UNIVERSITÀ DEGLI STUDI DI PADOVA

Department of Civil, Environmental and Architectural Engineering

Master's Degree in Mathematical Engineering

Mathematical Modelling for Engineering and Science

Master's Thesis

A mixed Material Point Method for the Stokes equations:
a stabilized velocity-pressure formulation

Supervisor:
Prof. Antonia Larese
University of Padua

Co-supervisor:
Prof. Laura Moreno Martinez
University of Alicante

External Supervisor:
Dr. Veronika Singer
Technical University of Munich

Candidate:
Giovanni Campanella
n. 2088283

Academic Year 2024/2025

Abstract

The Material Point Method (MPM) is a particle-based numerical technique, usually chosen for its ability to efficiently handle large deformation problems; for this reason, it has historically been used in a variety of applications within the field of solid mechanics and geomechanics. However, it has also been employed successfully in the field of fluid dynamics, with numerous recent developments available in the literature. This thesis proposes a novel MPM formulation for fluid dynamics problems. As the governing equations are solved using a Lagrangian description, this translates into numerically solving the Stokes problem using a Galerkin formulation. In particular, a mixed velocity-pressure formulation has been developed, as it's the most convenient approach to solve the Stokes equations. Mixed formulations in an MPM setting have been employed in several other works, both in solid mechanics and in fluid dynamics; it should be noted, however, that mixed velocity-pressure formulations like the one presented here have yet to be seen in the MPM literature. Well-known instability issues arise in mixed formulations of the Stokes problem due to the violation of the discrete *inf-sup* condition. To address this, various stabilization techniques have been developed. Among them a Variational-Multiscale (VMS) approach has been adopted here. The developed numerical scheme has been implemented within the *KRATOS Multiphysics* framework, and validated using the Method of Manufactured Solutions (MMS).

Keywords:

Material point, MPM, Mixed formulation, Stokes, Stabilized method

Aknowledgements

This development of this thesis has been largely carried out under the Technical University of Munich (TUM), through an exchange program sponsored by the Erasmus+ Traineeship program; the internship lasted from 07/10/2024 to 28/02/2025, and the hosting institution within the University was the Chair of Structural Analysis.

A special thanks therefore goes out to the Director of the Chair Prof. Dr.-Ing. habil. Roland Wüchner, to my local supervisor Dr. Veronika Singer, and to all other members of the Chair, who hosted me as one of their own and who provided ample assistance, guidance and support throughout this period, as well as to my colleague Andrea Morini, who joined me on this internship and with whom many fruitful ideas have been exchanged.

My utmost gratitude goes, in particular, to my supervisors, Prof. Antonia Larese and Prof. Laura Moreno, for the time and effort they dedicated in guiding me through the development of this thesis, helping me unconditionally in the earliest conceptualization phases, in the writing and debugging of the code, and in the drafting of this document.

Contents

Chapter 1

Introduction

1.1 Context and Motivation

Simulating fluid dynamics is a topic of great interest in the field of computational mechanics and engineering [1]; this study focuses on techniques to model fluid flows, and in particular on large-scale deformations of the flow domain in incompressible regimes. Several difficulties have to be overcome when modeling such flows: unless dedicated techniques are employed, numerical simulations in incompressible regimes are subject to pressure oscillations when the fluid is incompressible [2].

The most intuitive way to state the equations governing incompressible fluid flows without incurring in pressure instabilities is the mixed approach, where the velocity and pressure fields are coupled and solved by the numerical method as primary variables [3]; this translates into numerically solving the Stokes equation using the Galerkin method. However, mixed formulations may suffer from numerical instability issues related to the violation of the *inf-sup* condition [2]; a wide number of stabilization techniques have been formulated to fix this.

While the study of fluid dynamics has always been one of the topics of interest of MPM [4], its applications in the field of incompressible fluid flows are less fleshed out. A popular approach to model incompressible regimes is by using irreducible formulations, with just one kinematic variable (usually velocity) as the unknown; solving the resulting system of equations involves the use of the

fractional-step method [5], which may, however, suffer from instabilities related to the imposition of pressure boundary conditions [6]. While mixed formulations don't carry this drawback, their use in MPM hasn't been explored in-depth, with some notable examples being the leading work if Iaconeta et al. [7] in the field of incompressible solid mechanics, and the recent works of Moreno et al. [8] and Chandra et al. [9] treating hyperleastic materials and fluid flows respectively.

All the aforementioned mixed implementations adopt a displacement-pressure ($\boldsymbol{u}$-p) formulation; this is a common choice, since the elliptic problem describing incompressible, static and infinitesimal strains in solid mechanics is mathematically equivalent to the Stokes problem [10]). In fluid mechanics, however, a $\boldsymbol{v}$-p formulation would be favoured as the resulting weak formulation is described by linear operators only.

This work proposes a novel numerical scheme for fluid dynamics that addresses both complications related to highly distorted, incompressible fluid flows; a stabilized $\boldsymbol{v}$-p mixed formulation has been implemented with the Material Point Method (MPM), due to the ability of the former to model incompressible flows, and of the latter to prevent excessive mesh deformations, an issue that may be present in popular mesh-based methods like the Finite Element Method (FEM). In this introductory section, a general overview of the techniques that have been developed for this purpose will be provided, before going into greater detail on the specifics of the work presented in this thesis.

1.2 State of the Art

1.2.1 Numerical Methods for Fluid Dynamics

A wide variety of numerical techniques have been developed for the purposes of simulating fluid dynamics [1]. Such numerical schemes may be classified based on their discretization technique: mesh-based methods discretize the problem's domain into elements, by introducing a computational mesh that describes the fluid, while mesh-free methods approximate the fluid as a finite set of moving particles.

Mesh-based Methods

Some of the most proficient numerical schemes figure among the mesh-based methods, including the Finite Element Method (FEM), the Finite Difference

Method (FDM) and the Finite Volume Method (FVM). These methods may either adopt an Eulerian or a Lagrangian frame of reference, based on how their grids describe the fluid's motion.

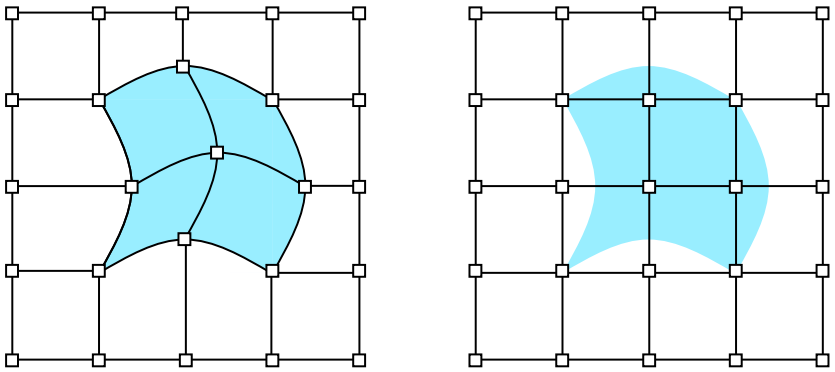


Figure 1.1: Comparison between Lagrangian (a) and Eulerian (b) methods.

Eulerian methods. In an Eulerian framework, the coordinate system is fixed in space; the fluid thus moves through the cells of a fixed computational grid. The fluid's properties may be stored in the mesh nodes or in the cell centers, based on the chosen numerical scheme. Eulerian methods prove to be fitting in situations of confined fluid flow; as the mesh does not distort as the fluid flows. However, convection of the physical quantities between cells during the fluid's motion may be a source of inaccuracy; this is particularly problematic when history-dependent quantities are involved. Finer structures within the fluid are also difficult to resolve. Moreover, this framework lacks an innate description of evolving free surfaces; this issue is commonly addressed by employing the level set method, a technique for surface tracking [11]. It does, however, require the solution of a re-initialization equation on the Eulerian grid, contributing to the computational burden.

Lagrangian methods. In a Lagrangian framework, the coordinate system follows the fluid itself; the cells encompassing the fluid will then move and deform along with it. A clear advantage of Lagrangian methods over its counterpart is the lack of convection terms in the governing equations, which may otherwise introduce numerical diffusion. Lagrangian frameworks perform best when accurate fluid boundaries are of interest (free-surface flows or multiple

fluids flows); mesh-based methods, however, usually fail to produce accurate predictions when great mesh deformations are involved, since the resulting cell distortion may compromise the solution leading to a loss of accuracy, or requiring unreasonably small timesteps. At worst, if the Jacobian of the deformed element turns negative, the element is inverted and the solution it yields will be non-physical.

Numerous mesh-based methods have been developed to overcome the issue of large mesh distortions in Lagrangian approaches. Remeshing techniques have been introduced in the Particle Finite Element Method (PFEM) [12]; this method employs a periodical rebuilding of the mesh to address the shortcomings of purely Lagrangian methods. PFEM is very effective at solving incompressible flows, and can readily handle complex problems such as moving free-surfaces, fluid-structure interactions and breaking waves. It does, however, come at the cost of a significantly increased computational effort, and at a potential loss of conservation properties, due to the repeated rebuilding of the mesh.

Hybrid Eulerian-Lagrangian frameworks also exist, such as the Arbitrary Lagrangian-Eulerian method (ALE) [13]. In this formulation, the extent of the mesh deformations can be prescribed to be limited; exceedingly high displacements can, however, still lead to excessive mesh deformations, similarly to Lagrangian methods. Moreover, the presence of convective terms is still an issue. In a similar vein, the immersed boundary (IB) method [14] features a mixture of Lagrangian and Eulerian variables, obtained by having a fixed Cartesian mesh (where the Eulerian variables are defined) and a Lagrangian description of the fluid; in this case, however, the Eulerian and the Lagrangian grids are not constrained to each other in any way at all.

Mesh-free Methods

Mesh-free methods are highly effective at simulating fluid flows. These methods work by discretizing the fluid into particles, and their movement is dictated by their interaction with neighbouring particles. They most often employ Lagrangian frameworks, and as such are exempt of the the numerical diffusion characterizing the Eulerian frame of reference, but they are also not afflicted by the distortion issues typical of Lagrangian meshes.

Among them, the most widespread is the Smoothed Particle Hydrodynamics (SPH) [15, 16]. First created to solve astrophysics problems, it's been subject of

numerous improvements over the years, leading to it becoming one of the most adopted solutions in computational fluid dynamics (CFD) [17, 18]. This method employs a fully Lagrangian description of the fluid particles, thus featuring all its related benefits while circumventing the shortcomings afflicting Lagrangian meshes; this choice allows it to be a very versatile approach for modeling fluid dynamics, being able to accurately simulate complex, nonlinear problems, even when multi-fluid flows and wall interactions are involved. Several version of SPH have been developed, including weakly-compressible and fully incompressible approaches [17]. Despite its clear advantages, SPH does have its share of drawbacks as well [17]: first of all, its computational cost is remarkably high, especially when 3D simulations are involved; moreover, it tends to produce poor predictions in certain scenarios, such as long-distance wave propagation, especially when it comes to pressure prediction; the prescription of boundary conditions is also quite challenging, as they cannot be directly included in its formal definition. While it can be shown, in many situations, to be as efficient as other well-established methods such as FEM, the mathematical theory underlying its convergence and stability is also not as clear.

Hybrid methods

Several attempts to merge the Lagrangian and Eulerian viewpoints into a hybrid framework have been performed over the years, aiming to exploit the advantages of both; this result is achieved by combining mesh-based and meshless methods. The Material Point Method, adopted in this thesis, is one of the latest developments among hybrid techniques. In the paragraphs to come, two of these methods will be described, as they were foundational to the creation of the MPM: the Particle-in-Cell method, and the Fluid Implicit Particle method.

Particle-in-Cell. The first Particle-in-Cell (PIC) method was developed in the late 1950s by Harlow [19] as a tool to solve highly distorted fluid flows in two dimensions. The method works by subdividing the computational domain in multiple cells; the fluid is represented in a Lagrangian frame by mass-carrying particles, which move through these cells. The partial differential equations describing the fluid's motion are discretized within these cells using the finite difference method. Other properties of the fluid are not stored in the particles themselves, and they are assigned to them proportionally to their mass.

The hybrid structure of PIC is achieved by having both an Eulerian background

mesh representing the computational domain, and a Lagrangian mesh modeling the fluid. In particular, each node of the Lagrangian mesh corresponds to a "particle" of fluid, where masses and positions are stored; velocity, internal energy and total mass are instead kept in the Eulerian grid. The fluid's motion is given by the flow of the Lagrangian particles through the Eulerian background mesh. Note that modeling convection is not required, as it is innately introduced by the movement of the particles through the computational grid.

Some of the advantages inherited from the Lagrangian framework include the ability to easily track the free-surface and history-dependent properties, while the Eulerian background grid prevents breakdowns due to large distortions. On the other hand, the main drawbacks PIC methods include the significant computational diffusion, and the excessive energy dissipation [20, 21]; the former is caused by transferring the momentum from the grid to the particles, then back to the grid, when convection is simulated through the grid, while the latter is caused by the inability to resolve small-scale structures.

Fluid Implicit Particle. The shortcomings of Harlow's PIC method have been addressed by Brackbill and Ruppel in 1986, through the development of the Fluid Implicit Particle (FLIP) method [20], which was later extended to incompressible fluid flows as well [21]. In FLIP, the issue of numerical diffusion is addressed by storing all the information in the particles; in particular, by storing energy and momentum there, the source of dissipation related to information transfer is greatly reduced.

Since in this method the information stored in the particles is enough to characterize the fluid flow, the background grid can be chosen more flexibly and, in particular, adaptive grids can be adopted to resolve finer details too, thus solving the issue of energy dissipation. Moreover, as the grid does not carry permanent information, it can be discarded and reconstructed after each timestep.

FLIP also introduced the distinction, within the computational cycle, between a Lagrangian and a convective phase; this allows for a greater choice of solution algorithm to be picked, including implicit methods, which prove to be most effective in many CFD applications.

Another crucial difference between FLIP and PIC lies in how the velocity is transferred from the grid back to the particles at the end of each time-step, with PIC interpolating the particle velocities directly from the grid velocities,

and FLIP deriving them from the grid accelerations instead. This choice carries better energy conservation properties, but it's known to be less stable than PIC's transfer scheme [22, 23].

Hybrid PIC/FLIP. In recent times, a hybrid PIC/FLIP method has been widely used in physics-based fluid animation [24, 25, 26]. This solution employs a staggered grid, a technique developed by Harlow in 1965 for the marker and cell (MAC) method [27]; in a staggered grid, some variables (such as the pressure) are stored in different locations, offset compared to the others. While this approach is almost free of numerical dissipation, and is generally more stable, it suffers from issues related to the staggered grid, which needs additional computational effort to be handled.

Material Point Method. The Material Point Method (MPM) was first developed by Sulsky et al. in 1994 [28] to efficiently treat continuum solid mechanics in large deformation regimes. Despite this being its original purpose, this method lends itself well to solve fluid dynamics problems too, with the first developments in this field appearing soon after its inception [4]. Indeed, MPM can trace back its roots to PIC and FLIP. Since these methods were originally developed in a CFD setting, employing the Material Point Method for similar applications can be highly effective. Inheriting its structure from its predecessors, MPM employs a hybrid Eulerian-Lagrangian framework, which grants it the necessary flexibility to treat large deformation problems, as well as highly distorted fluid flows. Storing material parameters in each particle allows this method to be proficient at dealing with complex, history-dependent constitutive laws, and as such MPM is greatly appreciated in the field of geotechnical engineering. An extensive review of the Material Point Method is given in chapter 3 of this thesis.

1.2.2 Mixed Formulations

The Stokes problem describes the incompressible flow of viscous fluids; its unknowns are velocity and pressure, and thus a mixed approach is adopted, solving the problem for both variables.

A linear system of equations arises from the discretization of the weak formulation of the Stokes equations; due to the saddle-point nature of the problem, its stability will be strongly related to the interpolation technique used to dis-

cretize the solution fields, as will be explained in further detail in Chapter 2. In particular, velocity and pressure must uphold a condition known as the Ladyzhenskaya-Babuška-Brezzi (LBB) or *inf-sup* condition, which states that the function space where the velocity solutions lie must be sufficiently "richer" than the pressure function space.

This condition can be proven to be violated by most commonly adopted solution spaces, including first-order spaces on both velocity and pressure ($\mathcal{P}_1 - \mathcal{P}_1$ spaces). Early attempts at stabilization include the adoption of higher-order spaces for the velocity only, by using *Taylor-Hood* spaces [29], that is, $\mathcal{P}_k - \mathcal{P}_{k-1}$ spaces with $k \geq 2$.

Complexities when dealing with basis functions of different orders for the two dependent variables, especially if of order higher than linear; several stabilization techniques have thus been developed, with the purpose of allowing the adoption of linear basis functions while guaranteeing stability; among them figure the isoparameteric elements and the bubble elements.

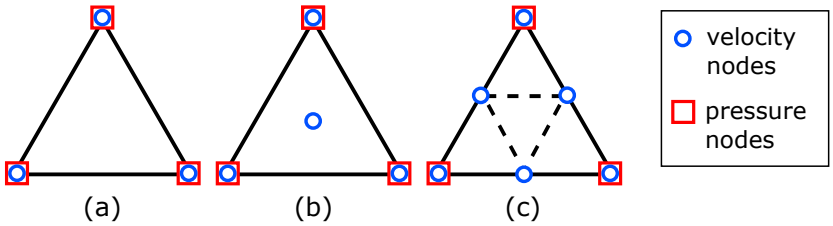


Figure 1.2: Mixed element types: (a) unstabilized element, (b) MINI element, (c) Isoparametric element.

Isoparametric elements $(\mathcal{P}_1 - \text{iso} - \mathcal{P}_2) - \mathcal{P}_1$ [30] stabilize the solution by adding a virtual node in the middle of each edge of the triangle $\mathcal{T}_{2h}$, thus creating a fictional triangle $\mathcal{T}_h \subset \mathcal{T}_{2h}$; the original, coarse triangulation $\mathcal{T}_{2h}$ is used to describe the pressure field, while the fictional, richer triangulation $\mathcal{T}_h$ is adopted for the velocity field. Note that, since $\mathcal{T}_h \subset \mathcal{T}_{2h}$, both solution fields can be expressed in terms of the finer one.

Bubble elements, also known as MINI elements $(\mathcal{P}_1 - \text{bubble}) - \mathcal{P}_1$ [31], enrich the standard $\mathcal{P}_1 - \mathcal{P}_1$ space with a bubble function β_T, with $\beta_T = 1$ in the center of gravity of the element, and zero at the boundary of the element.

An alternative approach to stabilization is the introduction of an additional term

$\boldsymbol{C}$ to the lower-right block of matrix (**??**), so that $\mathcal{A}$ becomes invertible; while adding a fictional term means committing an error (variational crime), $\boldsymbol{C}$ should be such that $\boldsymbol{C} \to 0$ when the numerical solution approaches the analytical one, in order to have an admissible variational crime and thus a consistent scheme.

The Galerkin Least-Square (GLS) method [32] works by adding the least-square form of the problem's residual to the Galerkin method. This results in a negative-definite matrix $\boldsymbol{C}$, thus preserving the saddle-point nature of $\mathcal{A}$ while stabilizing it.

A robust technique to stabilize the system matrix is the Variational Multiscale Method (VMS) [33], which similarly adds a stabilizing term $\boldsymbol{C}$ to the system matrix, but does so by modeling the infinitesimal scales of the solution that would otherwise be lost by numerical approximations. Several MPM formulations have chosen this method [8, 9], and it will thus be employed in this work; more details on its implementation are available in 2.7.1.

1.3 Research Gaps

The research behind this thesis was driven by a recorded gap in the research concerning implicit, velocity-based fluid flow simulations using the Material Point Method. Despite fluid dynamics having always been a topic of interest for MPM researchers [4], and incompressible flows being treated in more recent times in the field of soil-water interactions [34], there is a notable lack of research on the topic of free-surface incompressible flows. Existing formulations most often opt for irreducible approaches, either by imposing weak compressibility constraints [35] or employing fractional-step solution techniques [5]; on the other hand, current mixed formulations adopt displacement and pressure as unknowns [9], despite it being an uncommon choice in fluid dynamics. This work aims to fill this gap by proposing an implicit, velocity-based mixed formulation to solve the Stokes flow problem.

1.4 Objectives and Scope

1.4.1 Scientific Objectives

1. **Development of the v-p mixed numerical scheme:** this process involves stating the strong formulation of the problem by introducing the ap-

propriate conservation and constitutive laws which govern the incompressible fluid flow, and then obtaining a weak formulation using the Galerkin Finite Element method. The weak formulation then has to be discretized, both in space and in time, to obtain an algebraic system that can be solved numerically.

2. **Stabilization of the mixed formulation:** in order to address the arising oscillations in the solution fields, a detailed study of the root cause of this instability is required. The chosen stabilization method is the Variational Multiscale (VMS) method.

3. **Revision of the state-of-the-art of the Material Point Method:** as mentioned previously, mixed MPM formulation are scarce, particularly in the field of fluid dynamics; a detailed review of the inner workings of the Material Point Method is thus provided, followed by a review of current applications and techniques in the setting of fluid dynamics. The end goal is to clarify the reason why one may want to adopt this approach over other popular alternatives.

4. **Developement of a mixed MPM algorithm:** MPM can be seen as a Finite Element method where the integration points are allowed to move; the previously obtained numerical scheme is thus enhanced with the additional steps required by an MPM formulation, and the algorithm to be implemented is stated.

5. **Implementation:** the algorithm is implemented in the multiphysics framework *Kratos Multiphysics*. Since the software doesn't support velocity-based simulations in MPM, extensive additions are required for it to work; due to its layered and modular structure, interventions are required on several layers of the framework, including the user interface, the Python outer layer and the C++ inner classes.

6. **Testing and validation:** The quality and correctness of the implemented numerical scheme is evaluated by comparing the solutions it produces with known analytical solutions.

1.4.2 Methodology Overview

In this thesis, both analytical and numerical approaches have been employed to obtain the desired results: a preliminary mathematical analysis of the governing

equations of the Stokes problem has first been performed, leading to the development of a new numerical formulation, which was then validated numerically, by implementing it into *Kratos Multiphysics*. The ability of the newly written code to produce convergent results with progressively finer meshes has been used as the discerning criterion for the correctness of the method, according to known theoretical notions of numerical convergence.

1.5 Thesis Structure

The contents of this thesis will be organized as follows. In Chapter 2, a mixed Finite Element formulation of the Stokes problem is developed, providing an in-depth description of all the steps required, starting from the governing equations of the problem, stated in their strong formulation, before developing the corresponding weak formulation and discretizing its solution space both in space and in time. The underlying theory and the technique employed to stabilize the resulting linear system of equations is also explained.

Chapter 3 starts with a comprehensive revision of the state of the art of the Material Point Method, showcasing its structure and use cases and comparing it to other modern numerical methods in order to justify its adoption in this work; this section of the thesis focuses, in particular, on its adoption in the field of fluid mechanics. This review is then followed by the steps required to enrich the Finite Element formulation presented in the previous chapter, in order to obtain the mixed MPM formulation that is object of this study. An algorithm is provided, in order to guide the implementation of a dynamic scheme.

Chapter 4 describes the implementation of a static version of the scheme in *Kratos Multiphysics*; a description of the multi-layered, object-oriented structure of the software is first provided, with the purpose of explaining which sections of the code had to be tweaked or expanded in order to support the newly added algorithm.

In Chapter 5 the results of a variety of tests are reported, with the purpose of evaluating the method's convergence and accuracy under different scenarios, as well as the effectiveness of the employed stabilization technique.

Lastly, some concluding remarks to summarize the contents of this work are stated in Chapter 6, while also providing an outlook on future avenues of research that may be pursued starting from this thesis.

Chapter 2

Stokes Flow Problem

The evolution in time of a continuum is described by a boundary value problem, that is, a set of differential equations subject to boundary conditions and initial conditions. In particular, the governing differential equations in the setting of solid and fluid mechanics consist of a momentum balance equation, a mass balance equation and an equation describing the constitutive law of the continuum. An energy balance equation should, in principle, also be included; since, for the purposes of this thesis, thermal effects are neglected, this equation can, however, be implicitly assumed to be fulfilled [1]. This problem can be solved numerically by employing the Galerkin method [36, 37].

Within the context of fluid dynamics, the momentum balance is expressed through the Navier-Stokes equations, obtained by introducing the constitutive law of the Newtonian fluid; these equations are well-known for the nonlinearities associated with its convective terms, which require additional steps when solving them with numerical methods; the Material Point Method addresses this criticality by adopting a Lagrangian frame of reference, where no convective terms appear [28].

The Navier-Stokes equation may be stated in multiple different ways, each carrying their own draws and negative aspects. Two main choices to consider are whether to solve them for displacement or velocity, and whether to adopt an irreducible formulation (featuring only one of the two primary variables mentioned above) or a mixed formulation (using two primary variables by also introducing the pressure).

The choice of which primary variable to use depends on the nature of the problem. In the MPM setting, the irreducible, displacement-based approach is the most commonly used one, owing to the fact that the method was born for, and is primarily used in, solid mechanics applications [28]. In this field, formulations employing displacements are favored, as these problems mainly deal with deformations, which are expressed as a function of displacements.

On the other hand, velocity-based approaches are more common in computational fluid dynamics (CFD), as its equations are more intuitively expressed this way. It is not uncommon, however, to find displacement-based MPM schemes for fluid dynamics: due to its heritage being rooted in solid mechanics, this choice may be more convenient, as it allows to easily expand previous MPM developments in solid mechanics to CFD applications [9]. Moreover, this approach may be beneficial when implementing multi-phase problems: as will be better described in Section 3.4.1, such problems are implemented in MPM by assigning different constitutive laws to the particles representing each material; stating the momentum balance equations in a displacement-based fashion would thus allow to more easily account for the constitutive laws of the solid phase.

When choosing between an irreducible or a mixed formulation, the main features of the simulated phenomenon should be evaluated: near-incompressible or incompressible materials suffer from performance issues related to instabilities when handled with irreducible formulations [2, 37]. Mixed formulations were, at first, developed for CFD problems in the context of Finite Element simulations, where incompressible regimes are frequent. In the context of MPM, mixed formulations are have mainly been adopted for incompressible solid mechanics [7, 8], as the arising $\boldsymbol{u} - p$ equations are mathematically equivalent to the $\boldsymbol{v} - p$ Stokes problem in fluid dynamics [38], which is readily handled with a mixed approach.

On the other hand, the topic of mixed MPM schemes for fluid dynamics is significantly less fleshed out; a mixed displacement-pressure formulation has been developed by Chandra et al. [9], though this choice of primary variables meant that the resulting weak formulation had to be linearized, a step that is not needed when using a velocity-pressure formulation.

It should be noted that problems in incompressible regimes don't necessarily need to be solved using a mixed approach: alternative strategies include, for example, weak compressibility [35] and the fractional-step method [5, 39]. Both

approaches are, however, subject to instability, due to volumetric locking in the former case, and due to the necessity to impose pressure boundary conditions in the latter.

While a mixed formulation is the most natural approach to handle a CFD problem, it is not exempt of issues either. Notably, under most circumstances mixed schemes fail to satisfy the Ladyzhensaya–Babuska–Brezzi (LBB) or *inf-sup* condition for stability [2], leading to oscillations in the pressure; this topic will be delved into in much greater detail later in this section. There are, however, multiple stabilization techniques available in the literature, that allow to circumvent this issue.

The arguments provided above suggest the stabilized, mixed $\boldsymbol{v} - p$ formulation to be a very convenient approach to solve the Stokes problem, and it will thus be the one this thesis will follow.

In this section, all the necessary steps to develop a numerical scheme for the Stokes problem using the Galerkin method will be explained in detail, starting from the strong formulation to obtain the mixed weak formulation; the algebraic system to be solved numerically is then obtained through spatial and temporal discretizations. Finally, to avoid unphysical oscillations in the pressure, the Variational Multiscale (VMS) stabilization method will be showcased, after an introduction on the root causes of instability in mixed problems.

2.1 Strong Formulation

Definition 2.1.1 (Reference configuration)**.** Let $\mathcal{E} = \mathbb{R}^d$ be a d-dimensional Euclidean space, with $d = 1, 2, 3$; let $\Omega \subset \mathcal{E}$ be a region in this space, and $\partial\Omega$ be its boundary; Ω is taken as an open and connected set with locally Lipschitz continuous boundary, and outer unit normal $\boldsymbol{n}$ defined almost everywhere.

Let $\mathcal{B}$ be a body or fluid mass consisting of an infinite set of points $\boldsymbol{X}$ occupying this region; these points provide a description of the material or reference configuration of this body. □

Definition 2.1.2 (Updated configuration)**.** Let φ be a one-to-one mapping

$$\varphi : \Omega \to \mathcal{E}, \tag{2.1}$$

where each point $\boldsymbol{X}$ of the reference configuration is mapped into a point $\boldsymbol{x}$ over

a timeframe $]0, T[$, according to:

$$\boldsymbol{x} = \varphi(\boldsymbol{X}, t), \quad \forall \boldsymbol{X} \in \Omega, \forall t \in]0, T[; \tag{2.2}$$

$\varphi(\Omega, t)$ then represents the spatial or updated configuration, that is, the region occupied by $\mathcal{B}$ over time, as it undergoes a deformation or displacement. The boundary of this configuration is $\varphi(\partial\Omega, t)$. □

Definition 2.1.3 (Space-time domain)**.** The space-time domain of a problem is defined as:

$$\mathfrak{D}_s = \{\boldsymbol{x} | \boldsymbol{x} \in \varphi(\Omega, t), 0 < t < T\} \tag{2.3}$$

□

Definition 2.1.4 (Material derivative)**.** Let $\boldsymbol{y}(\boldsymbol{x}, t)$ be a vector field which depends on the space $\boldsymbol{x}$ and the time t. Let $\boldsymbol{v}(\boldsymbol{x}, t)$ be a space- and time-dependent velocity field. The material derivative is defined as:

$$\frac{d\square}{dt} \coloneqq \frac{\partial \square}{\partial t} + \boldsymbol{v} \cdot \nabla(\square). \tag{2.4}$$

□

Remark 2.1.5. Assuming a Lagrangian frame of reference (as will be done in the present formulation), the material derivative can be simplified to neglect the additional convection term, such that $d/dt \equiv \partial/\partial t$.

Definition 2.1.6 (Kinematic quantities)**.** The displacement $\boldsymbol{u}(\boldsymbol{x}, t)$ is defined as the difference between the updated and the reference configuration:

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{x}(\boldsymbol{X}, t) - \boldsymbol{X} \tag{2.5}$$

The velocity $\boldsymbol{a}(\boldsymbol{x}, t)$ and acceleration $\boldsymbol{v}(\boldsymbol{x}, t)$ are defined as:

$$\boldsymbol{v}(\boldsymbol{x}, t) = \frac{d\boldsymbol{u}(\boldsymbol{x}, t)}{dt} \tag{2.6}$$

$$\boldsymbol{a}(\boldsymbol{x}, t) = \frac{d\boldsymbol{v}(\boldsymbol{x}, t)}{dt} \tag{2.7}$$

□

Definition 2.1.7 (Momentum balance equation)**.** Let $\rho(\boldsymbol{x}, t)$ be the density, $\boldsymbol{a}(\boldsymbol{x}, t)$ be the acceleration, $\boldsymbol{\sigma}(\boldsymbol{x}, t)$ be the stress tensor and $\boldsymbol{f}(\boldsymbol{x}, t)$ be the body

force.

The momentum balance equation is:

$$\rho(\boldsymbol{x},t)\boldsymbol{a}(\boldsymbol{x},t) = \nabla \cdot \boldsymbol{\sigma}(\boldsymbol{x},t) + \boldsymbol{f}(\boldsymbol{x},t), \tag{2.8}$$

□

From now on, in order to lighten the notation, the explicit variables $(\boldsymbol{x},t)$ of each unknown term will be omitted.

Definition 2.1.8 (Mass balance equation)**.** The mass balance equation is:

$$\frac{d\rho}{dt} + \rho\nabla \cdot \boldsymbol{v} = 0 \tag{2.9}$$

□

The boundary conditions are applied on $\varphi(\partial\Omega)$; in particular, the boundary is split in two disjoint parts, $\partial\Omega_D$ and $\partial\Omega_N$, where the Dirichlet and Neumann boundary conditions, respectively, are applied. These constraints are enforced as:

$$\boldsymbol{v} = \boldsymbol{v}_D \qquad \text{on } \varphi(\partial\Omega_D) \tag{2.10}$$

$$\boldsymbol{n} \cdot \boldsymbol{\sigma} = \boldsymbol{t}_N \qquad \text{on } \varphi(\partial\Omega_N) \tag{2.11}$$

where $\boldsymbol{v}_D$ and $\boldsymbol{t}_N$ are the enforced values of the velocity and the traction at the respective boundary.

To obtain the mixed velocity-pressure formulation for an incompressible Newtonian fluid, its constitutive law is introduced.

The constitutive law of any moving fluid is dictated by empirical observations, through which it is known that, on a given element of fluid, the tangential stresses are non-zero and the normal stress is not isotropic. For this reason, the stress is decomposed into an isotropic, hydrostatic scalar quantity, and the remaining non-isotropic, deviatoric component:

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + \boldsymbol{\sigma}^{dev}. \tag{2.12}$$

$\boldsymbol{I}$ is the d-dimensional identity matrix, and the scalar quantity p is the dynamic

pressure, which is given by the average of the d normal stresses σ^{hyd} (which are, in principle, different from each other due to their anisotropy):

$$p = -\frac{\sum_{i=1}^{d} \sigma_{i,i}^{hyd}}{d}; \tag{2.13}$$

it should be noted that the dynamic pressure is not the same value as the static pressure that is measured, for example, in a fluid at rest ($\boldsymbol{v} = 0$).

$\boldsymbol{\sigma}^{dev}$ contributes to the tangential stresses, and its definition is related to the nature of the fluid.

Definition 2.1.9 (Newtonian fluid)**.** A Newtonian fluid is a fluid that fulfills the following properties [40]:

- $\boldsymbol{\sigma}$ is a linear and continuous function of the strain rate tensor $\dot{\boldsymbol{\varepsilon}}$, which is defined as
$$\dot{\varepsilon} = \frac{1}{2}\left(\nabla \boldsymbol{v} + \nabla^T \boldsymbol{v}\right) \coloneqq \nabla^s \boldsymbol{v}; \tag{2.14}$$
- $\boldsymbol{\sigma}$ does not depend on $\boldsymbol{x}$ explicitly (homogeneous fluid), but only through the strain rate tensor $\dot{\boldsymbol{\varepsilon}}$;
- the fluid is isotropic;
- if $\dot{\boldsymbol{\varepsilon}} = 0$, the stress is also isotropic ($\boldsymbol{\sigma} = -p\boldsymbol{I}$).

□

Definition 2.1.10 (Constitutive law of a Newtonian fluid)**.** Let μ be the dynamic viscosity and λ be the volumetric viscosity. The constitutive law of a Newtonian fluid is:

$$\boldsymbol{\sigma} = p\boldsymbol{I} + 2\mu\dot{\boldsymbol{\varepsilon}} + \lambda(\nabla \cdot \boldsymbol{v})\boldsymbol{I} \tag{2.15}$$

□

Incompressibility is then introduced in the mass balance (2.9), which thus simplifies to

$$\nabla \cdot \boldsymbol{v} = 0; \tag{2.16}$$

this equation states that the rate of expansion of the fluid is null, and the last term of (2.15) is therefore removed:

$$\boldsymbol{\sigma} = p\boldsymbol{I} + \ 2\mu\nabla^s \boldsymbol{v} \tag{2.17}$$

Substituting (2.17) in the momentum balance equation (2.8), the incompressible Navier-Stokes equation is obtained:

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v}\cdot\nabla\boldsymbol{v} - \mu\Delta\boldsymbol{v} + \nabla p = \boldsymbol{f} \tag{2.18}$$

As mentioned previously, adopting a Lagrangian frame of reference causes the nonlinear convective term $\boldsymbol{v}\cdot\nabla\boldsymbol{v}$ to vanish, leading to the Stokes equations:

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} - \mu\Delta\boldsymbol{v} + \nabla p = \boldsymbol{f} \tag{2.19}$$

Problem 2.1.11 (Strong formulation)**.** The strong formulation of the boundary value problem describing the dynamics of an incompressible Newtonian fluid, stated using a mixed velocity-pressure formulation, consists in finding a velocity field $\boldsymbol{v} : \mathfrak{D}_s \to \mathbb{R}^d$ and a pressure field $p : \mathfrak{D}_s \to \mathbb{R}$ such that:

$$\begin{cases} \rho\dfrac{\partial \boldsymbol{v}}{\partial t} - \mu\Delta\boldsymbol{v} + \nabla p = \boldsymbol{f} & \text{in } \varphi(\Omega,t), t \in]0,T[\\ \nabla\cdot\boldsymbol{v} = 0 & \text{in } \varphi(\Omega,t), t \in]0,T[\\ \boldsymbol{v} = \boldsymbol{v}_D & \text{on } \varphi, t \in]0,T[\\ \boldsymbol{n}\cdot\nabla^s\boldsymbol{v} = \boldsymbol{t}_N & \text{on } \varphi(\partial\Omega_N), t \in]0,T[\end{cases} \tag{2.20}$$

□

It should be noted, however, that while the boundary conditions do not involve the pressure, it must still be constrained, since in the governing equations it only appears under the gradient operator, and thus is only defined up to an arbitrary constant. For this reason, an additional constraint is added:

$$\int_{\varphi(\Omega)} p \, d\boldsymbol{x} = 0 \tag{2.21}$$

2.2 Weak Formulation

The strong formulation yields a solution $\boldsymbol{v}$ which satisfies the governing equations and, in order to be well-defined, requires continuity up to its second order derivative ($\boldsymbol{v} \in \mathcal{C}^2(\Omega)$). The weak form relaxes this requirement by looking for a solution within specific function spaces. Some insight on the topic will be given in this section; since the ultimate goal of this thesis is to present a

mixed approach, it is indeed crucial to provide a general overview of the weak formulation, as its well-posedness for the discrete mixed problem is not trivial, and a lack of understanding behind its theory has historically been source of misconceptions.

First, some introductory concepts have to be introduced, as they will be foundational for the later steps of this section.

In order to relax some of the continuity requirements, the derivatives have to be redefined as weak derivatives.

Definition 2.2.1 (Weak derivative)**.** Given two functions $u, v : \Omega \to \mathbb{R}$ and a multi-index α, $v \coloneqq D^\alpha u$ is the weak derivative of u if:

$$\int_\Omega v\phi \; d\boldsymbol{x} = (-1)^{|\alpha|} \int_\Omega u D^\alpha \phi \; d\boldsymbol{x} \qquad \forall \phi \in \mathcal{C}_c^\infty(\Omega) \tag{2.22}$$

where $\mathcal{C}_c^\infty(\Omega)$ are smooth functions with compact support. □

Remark 2.2.2. Using this definition, u needs not be differentiable everywhere; moreover, should $u \in \mathcal{C}^1(\Omega)$, this definition would coincide with that of the standard derivative.

The previously defined domain Ω is trivially Lebesgue measurable; functions defined in this domain are measurable ($v \in L(\Omega)$). Further notions on measurability will not be included, so as not to go out of the scope of this thesis.

The function space $L^p(\Omega)$ can now be defined.

Definition 2.2.3 (L^p space)**.** The function space $L^p(\Omega)$ is the space of p-integrable, measurable functions:

$$L^p(\Omega) = \left\{ v : \Omega \to \mathbb{R} : v \in L(\Omega), \int_\Omega |v(\boldsymbol{x})|^p \; d\boldsymbol{x} < +\infty \right\}, \tag{2.23}$$

where the integral is to be intended as a Lebesgue integral. □

Among the L^p spaces, L^2 is a Hilbert space, with scalar product

$$(v, w)_{L^2(\Omega)} = \int_\Omega v(\boldsymbol{x}) w(\boldsymbol{x}) \; d\boldsymbol{x} \tag{2.24}$$

and induced norm

$$||v||_{L^2(\Omega)} \coloneqq \sqrt{(v,v)_{L^2(\Omega)}} = \left(\int_\Omega v(\boldsymbol{x})^2 \; d\boldsymbol{x}\right)^{1/2}. \tag{2.25}$$

The space where the candidate solutions for the weak formulations are found is $H^1(\Omega)$.

Definition 2.2.4 (H^1 space)**.** $H^1(\Omega)$ is the space of square integrable functions with square integrable first derivatives (in the weak sense):

$$H^1(\Omega) = \left\{v : \Omega \to \mathbb{R} : v \in L^2(\Omega),\, D^\alpha v \in L^2(\Omega)\right\} \tag{2.26}$$

□

$H^1(\Omega)$ is a Hilbert space with scalar product

$$\begin{aligned}(v,w)_{H^1(\Omega)} &= \int_\Omega \left(v(\boldsymbol{x})w(\boldsymbol{x}) + \nabla v(\boldsymbol{x}) \cdot \nabla w(\boldsymbol{x})\right) \; d\boldsymbol{x} \\ &= (v,w)_{L^1(\Omega)} + (\nabla v, \nabla w)_{L^1(\Omega)}\end{aligned} \tag{2.27}$$

and induced norm

$$||v||^2_{H^1(\Omega)} = ||v||^2_{L^2(\Omega)} + ||\nabla v||^2_{L^2(\Omega)} \tag{2.28}$$

In the weak formulation, the functions in this space must vanish at the Dirichlet boundary $\varphi(\partial\Omega_D)$; the corresponding function space is $H^1_0(\Omega)$.

Definition 2.2.5 (H^1_0 space)**.** $H^1_0(\Omega)$ is the space of square integrable functions with square integrable first derivatives (in the weak sense) that vanish at the Dirichlet boundary $\varphi(\partial\Omega_D)$:

$$H^1_0(\Omega) = \left\{v \in H^1(\Omega) : v|_{\varphi(\partial\Omega_D)} = 0\right\} \tag{2.29}$$

□

This additional condition actually requires special care when imposing it: indeed, it is applied to functions that belong in L^2; however, since the boundary is a measure-zero set of $\mathbb{R}^d$, the integral that defines such function space is not well-defined on $\partial\Omega$.

It can, however, be proven that functions in Hilbert spaces can be approximated

as sequences of smooth functions.

Theorem 2.2.6. Let $\Omega \in \mathbb{R}^d$ be an open set, and $v \in H^k(\Omega)$. Then, there exists a sequence $\{v_j\}$ of smooth functions ($v_j \in \mathcal{C}^\infty(\Omega)$) such that $||v - v_j||_{H^k(\Omega)} \to 0$ (written as $v_j \to v$ in $H^k(\Omega)$).

Moreover, if $\partial\Omega$ is at least Lipschitz continuous, then there exists a sequence $\{v_j\}$ of smooth functions $v_j \in \mathcal{C}^\infty(\bar{\Omega})$ (functions that are regular up to the boundary) such that $v_j \to v$ in $H^k(\Omega)$. □

Using the above theorem, the space H_0^1 can be defined as the closure of $\mathcal{C}_c^\infty(\Omega)$ in H_0^1, that is, all the functions in H^1 that can be described as the limit, in the H^1 norm (2.28), of a sequence of functions in $\mathcal{C}_c^\infty$ (which vanish on $\partial\Omega$, having compact support).

The notion of $L^p(\Omega)$ spaces can be generalized to vector-valued functions $\boldsymbol{v} : \Omega \to \mathbb{R}^d$ (or, more generally, to functions $\boldsymbol{v} : \Omega \to E$, with E being a Banach space). Limiting this analysis to $L^2(\Omega, \mathbb{R}^d)$, the space will be equipped with the scalar product

$$(\boldsymbol{v}, \boldsymbol{w})_{L^2(\Omega,\mathbb{R}^d)} = \sum_{i=1}^{d} (v_i, w_i)_{L^2(\Omega)} = \int_\Omega \boldsymbol{v} \cdot \boldsymbol{w} \; d\boldsymbol{x} \tag{2.30}$$

and with the induced norm

$$||\boldsymbol{v}||^2_{L^2(\Omega,\mathbb{R}^d)} = \sum_{i=1}^{d} ||v_i||^2_{L^2(\Omega)} \tag{2.31}$$

Note that requiring a vector-valued function to be $L^2(\Omega, \mathbb{R}^d)$ is equivalent to requiring each of its components to be $L^2(\Omega)$, that is, $\boldsymbol{v} \in \left[L^2(\Omega)\right]^d$.

The scalar product and induced norm of $H_0^1(\Omega, \mathbb{R}^d)$ can be analogously derived and the procedure will thus be omitted. Moreover, for convenience, a more compact notation $(\square, \square)_{\mathbb{R}^d} \coloneqq (\square, \square)_{L^2(\omega,\mathbb{R}^d)}$ will be adopted from now on, with ω being the domain.

Definition 2.2.7 (L_0^2 space)**.** $L_0^2(\Omega)$ is the space of $L^2(\Omega)$ functions where the constraint (2.21) is applied:

$$L_0^2(\Omega) = \left\{ p \in L^2(\Omega) : \int_\Omega p \; d\boldsymbol{x} = 0 \right\} \tag{2.32}$$

Now that all the required spaces have been properly defined, the weak form can be stated. Let $\mathcal{V} = H_0^1(\varphi(\Omega), \mathbb{R}^d)$ and $\mathcal{Q} = L_0^2(\varphi(\Omega))$ be the function spaces, where the solution of the weak form will be sought, such that $\boldsymbol{v} \in \mathcal{V}$, $p \in \mathcal{Q}$. For practical purposes, the unknowns of the problem are expressed as

$$\boldsymbol{V} = \begin{pmatrix} \boldsymbol{v} \\ p \end{pmatrix} \in \mathcal{W} \tag{2.33}$$

with $\mathcal{W} = \mathcal{V} \times \mathcal{Q}$. The governing equations of the problem are also rewritten as:

Find $(\boldsymbol{v}, p) \in \mathcal{V}$ s.t.

$$\mathcal{D}_t(\boldsymbol{V}) + \mathcal{L}(\boldsymbol{V}) = \mathcal{F} \tag{2.34}$$

with:

$$\mathcal{D}_t(\boldsymbol{V}) = \begin{pmatrix} \frac{\partial \boldsymbol{v}}{\partial t} \\ 0 \end{pmatrix} \tag{2.35}$$

$$\mathcal{L}(\boldsymbol{V}) = \begin{pmatrix} -\mu \Delta \boldsymbol{v} + \nabla p \\ \nabla \cdot \boldsymbol{v} \end{pmatrix} \tag{2.36}$$

$$\mathcal{F} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix} \tag{2.37}$$

Moreover, introduce a generic vector of test functions belonging to $\mathcal{W}$:

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{w} \\ q \end{pmatrix} \in \mathcal{W} \tag{2.38}$$

Equation (2.34) is then multiplied by the test functions and integrated over $\varphi(\Omega)$:

$$(??)\, \langle \mathcal{D}_t(\boldsymbol{V}), \boldsymbol{W} \rangle_{\varphi(\Omega)} + \langle \mathcal{L}(\boldsymbol{V}), \boldsymbol{W} \rangle_{\varphi(\Omega)} = \langle \mathcal{F}, \boldsymbol{W} \rangle_{\varphi(\Omega)} \tag{2.39}$$

In the above notation, $\langle \square, \square \rangle_{\varphi(\Omega)}$ is the integral over $\varphi(\Omega)$ of the product of two generic functions. In particular:

$$\langle \mathcal{D}_t(\boldsymbol{V}), \boldsymbol{W} \rangle_{\varphi(\Omega)} = \left(\frac{d\boldsymbol{v}}{dt}, \boldsymbol{w} \right)_{\mathbb{R}^d} \tag{2.40}$$

$$\langle \mathcal{L}(\boldsymbol{V}), \boldsymbol{W} \rangle_{\varphi(\Omega)} = (-\mu \Delta \boldsymbol{v}, \boldsymbol{w})_{\mathbb{R}^d} + (\boldsymbol{w}, \nabla p)_{\mathbb{R}^d} + (-\nabla \cdot \boldsymbol{v}, q)_{\mathbb{R}} \tag{2.41}$$

$$\langle \mathcal{F}, \boldsymbol{W} \rangle_{\varphi(\Omega)} = \langle \boldsymbol{f}, \boldsymbol{w} \rangle_{\varphi(\Omega)} \tag{2.42}$$

Due to the velocity and pressure fields being described by an H^1 and L^2 space respectively, neither $\Delta \boldsymbol{v}$ nor ∇p are well-defined; Green's lemma is then applied to (2.41). From now on, homogeneous Neumann and Dirichlet boundary conditions will be assumed for simplicity, such that boundary terms will vanish.

$$(-\mu \Delta \boldsymbol{v}, \boldsymbol{w})_{\mathbb{R}^d} = (\mu \nabla^s \boldsymbol{v}, \nabla^s \boldsymbol{w})_{\mathbb{R}^{d\times d}} \tag{2.43}$$

$$(\boldsymbol{w}, \nabla p)_{\mathbb{R}^d} = (-\nabla \cdot \boldsymbol{w}, p)_{\mathbb{R}} \tag{2.44}$$

In (2.43), the scalar product $(\square, \square)_{\mathbb{R}^{d\times d}}$ is defined similarly as (2.30); however, the Frobenius inner product (a generalization of the vector dot product to matrices) is used instead of the dot product:

$$(\mu \nabla^s \boldsymbol{v}, \nabla^s \boldsymbol{w})_{\mathbb{R}^{d\times d}} = \int_{\varphi(\Omega)} \mu \nabla^s \boldsymbol{v} : \nabla^s \boldsymbol{w} \; d\boldsymbol{x}, \tag{2.45}$$

where $\boldsymbol{A} : \boldsymbol{B} = \mathrm{Tr}(\boldsymbol{A}^T \boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{A}\boldsymbol{B}^T)$ is the Frobenius inner product, also known as tensor double contraction.

Moreover, notice that a minus sign has been added to the last term of (2.42), by inverting the signs of the mass balance equation. As Green's lemma outputs a similar term in equation (2.44), this analogy will allow for a simpler description of the problem later on.

Now, rewrite the problem (2.41) in terms of bilinear forms and linear functionals.

$B(\boldsymbol{V}, \boldsymbol{W})$ is a bilinear form defined as:

$$B(\boldsymbol{V}, \boldsymbol{W}) \coloneqq (\mu \nabla^s \boldsymbol{v}, \nabla^s \boldsymbol{w})_{\mathbb{R}^{d\times d}} + (-\nabla \cdot \boldsymbol{v}, q)_{\mathbb{R}} + (-\nabla \cdot \boldsymbol{w}, p)_{\mathbb{R}} \tag{2.46}$$

In vector notation, $B(\boldsymbol{V}, \boldsymbol{W})$ is:

$$B(\boldsymbol{V}, \boldsymbol{W}) = \begin{pmatrix} a(\boldsymbol{v}, \boldsymbol{w}) + b(\boldsymbol{w}, p) \\ b(\boldsymbol{v}, q) \end{pmatrix} \tag{2.47}$$

where $a(\boldsymbol{v}, \boldsymbol{w}) = (\mu \nabla^s \boldsymbol{v}, \nabla^s \boldsymbol{w})_{\mathbb{R}^{d\times d}}$ and $b(\boldsymbol{w}, q) = (-\nabla \cdot \boldsymbol{w}, q)_{\mathbb{R}}$ are also bilinear forms.

$F(\boldsymbol{W})$ is a linear functional defined as:

$$F(\boldsymbol{W}) = \langle \boldsymbol{f}, \boldsymbol{w} \rangle_{\varphi(\Omega)} \tag{2.48}$$

Problem 2.2.8 (Weak formulation)**.** The weak formulation of the boundary value problem describing the dynamics of an incompressible Newtonian fluid, stated using a mixed velocity-pressure formulation, consists in finding $\boldsymbol{V} \in \mathcal{W}$ such that:

$$\langle \mathcal{D}_t(\boldsymbol{V}), \boldsymbol{W} \rangle_{\varphi(\Omega)} + B(\boldsymbol{V}, \boldsymbol{W}) = F(\boldsymbol{W}) \qquad \forall\ \boldsymbol{W} \in \mathcal{W} \tag{2.49}$$

2.3 Spatial Discretization

The spatial discretization is performed by partitioning the domain into a mesh, and assigning the appropriate basis functions to its nodes.

Definition 2.3.1 (Element)**.** An element (or cell) T is a compact polyhedron in $\mathbb{R}^d$ whose interior is not empty. Its boundary ∂T is formed by m-dimensional linear manifolds (points, segments, pieces of planes) called m-faces, with $0 \leq m \leq d-1$. The 0-faces are called nodes (or vertices), the 1-faces are called edges (or segments) and the $d-1$-faces are called faces. □

Definition 2.3.2 (Triangulation)**.** Let $\Omega \subset \mathcal{E}$ be a domain, with its closure $\bar{\Omega}$ having polygonal boundary $\Gamma = \partial\Omega$. A triangulation (or mesh, or grid) $\mathcal{T}_h(\Omega)$ is a partition of $\bar{\Omega}$ into N_T non-overlapping elements T_k, such that:

$$\mathcal{T}_h(\Omega) = \{\cup_{e=1}^{N_e} T_e = \bar{\Omega},\ T_i \cap T_j = \sigma_{ij}\} \tag{2.50}$$

where $\sigma_{ij} \subset \mathbb{R}^m$, $m = \{0, \ldots, d-1\}$ and h is a characteristic length parameter, which is defined as

$$h_K \coloneqq \mathrm{diam}(K), K \in \mathcal{T}_h, \qquad h = \max_{K \in \mathcal{T}_h} h_K \tag{2.51}$$

□

Definition 2.3.3 (Admissible triangulation)**.** A triangulation $\mathcal{T}_h(\Omega)$ is admissible if:

- each mesh element $T \in \mathcal{T}_h(\Omega)$ is closed and its interior T° is non-empty;

- the elements are non-overlapping, that is, $T_i^\circ \cap T_j^\circ \ \forall \ T_i, T_j, i \neq j$;
- the boundary of each $T \in \mathcal{T}_h$ is Lipschitz continuous;
- The intersection of two mesh elements is either empty or a common m-face, $m \in \{0, \ldots, d-1\}$

□

Theorem 2.3.4. Let $T \in \mathcal{T}_h$ be an element; let $\mathcal{P}_1(T)$ be the set of polynomials of degree at most 1 whose support is T. A function $v(\boldsymbol{x}) \in \mathcal{P}_1(T)$ is uniquely determined by its values at the vertices. □

To discretize the domain of this problem, an admissible triangulation $\mathcal{T}_h(\varphi(\Omega))$ with N_n nodes is defined.

Remark 2.3.5. In practice, the partition $\mathcal{T}_h(\varphi(\Omega))$ will not actually match $\varphi(\Omega)$; since this approximation is neglected, a variational crime is committed. This error, however, decreases as the mesh is refined.

Now that the mesh has been defined, the solution spaces are discretized by finding finite-dimensional subspaces where the numerical solution is sought.

Definition 2.3.6 (Discretized solution space)**.** Let $\mathcal{V}$ be a separable solution space, meaning that it can be described by a countable set of functions (basis functions), and let ϕ_i be its basis functions. The discretized solution space $\mathcal{V}_h$ is defined as the span of a finite subset of n basis functions:

$$\mathcal{V}_h = \text{span}\{\phi_1, \phi_2, \ldots, \phi_n\} \tag{2.52}$$

□

Assuming both $\mathcal{V}$ and $\mathcal{Q}$ are separable, and letting $\boldsymbol{\phi}$ and γ be their respective basis functions, the discretized solution spaces of the mixed problem are thus:

$$\mathcal{V}_h = \text{span}\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_{N_n}\} \tag{2.53}$$

$$\mathcal{Q}_h = \text{span}\{\gamma_1, \gamma_2, \ldots, \gamma_{N_n}\} \tag{2.54}$$

Functions in $\mathcal{V}_h$ and $\mathcal{Q}_h$ can then be obtained as a linear combination of the

basis functions, weighted with the numerical values on the nodes:

$$\boldsymbol{v}_h(\boldsymbol{x},t) = \sum_{i=1}^{N_n} \boldsymbol{v}_i(t)\phi_i(\boldsymbol{x}) \tag{2.55}$$

$$p_h(\boldsymbol{x},t) = \sum_{i=1}^{N_n} p(t)\gamma_i(\boldsymbol{x}) \tag{2.56}$$

Remark 2.3.7. In this work, linear basis functions will be adopted for both unknowns of the problem, and the notation ϕ_i will be used for pressure basis functions as well. Therefore, as stated in theorem (2.3.4), interpolating $\boldsymbol{v}$ and p on the N_n nodes of the triangulation is enough to uniquely determine it.

The linear basis functions are defined on each node N_i such that:

$$\phi_i(\boldsymbol{x}_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \qquad i,j = 1,\dots,N_n; \tag{2.57}$$

thus, they are piecewise linear functions with support defined by all triangles sharing the node N_i. The discretized spaces can thus also be written as

$$\mathcal{V}_h = \{\boldsymbol{v}(\boldsymbol{x}) : \boldsymbol{v} \in \mathcal{C}^0(\varphi(\Omega)),\, v_i|_{T_e} \in \mathcal{P}_1(T_e)\ \forall\ T_e \in \mathcal{T}_h,\, \boldsymbol{v} = 0 \text{ in } \varphi(\partial\Omega_D)\} \tag{2.58}$$

$$\mathcal{Q}_h = \left\{ p(\boldsymbol{x}) : p \in \mathcal{C}^0(\varphi(\Omega)),\, p|_{T_e} \in \mathcal{P}_1(T_e)\ \forall\ T_e \in \mathcal{T}_h,\, \int_{\varphi(\Omega)} p = 0 \right\} \tag{2.59}$$

Assuming $d = 2$, the element-wise $\mathcal{P}_1$ functions are given by:

$$\phi_i^{T_e}(\boldsymbol{x}) = \frac{\alpha_i^{T_e} + \beta_i^{T_e} x + \gamma_i^{T_e} y}{|T_e|} \tag{2.60}$$

In the Material Point Method, the computational domain is partitioned into elements, rather than the body, as will be later discussed more in detail. For the purposes of deriving the discretized weak formulation, however, the process is analogous.

Problem 2.3.8 (FEM problem)**.** Find $\boldsymbol{V}_h \in \mathcal{W}_h = \mathcal{V}_h \times \mathcal{Q}_h$ such that:

$$\langle \mathcal{D}_t(\boldsymbol{V}_h), \boldsymbol{W}_h \rangle_{\varphi(\Omega)} + B(\boldsymbol{V}_h, \boldsymbol{W}_h) = F(\boldsymbol{W}_h) \qquad \forall\ \boldsymbol{W}_h \in \mathcal{W}_h \tag{2.61}$$

□

2.4 Temporal Discretization

In equations (2.55)-(2.56), separation of variables has been performed on the solution fields: space- and time-dependent functions are separated such that the basis functions only depend on spatial variables while the values at the nodes only depend on time:

$$y_h(\boldsymbol{x}, t) = \sum_{i=1}^{N_n} y_i(t)\phi_i(\boldsymbol{x}) \tag{2.62}$$

To describe the evolution in time of the system, a time-stepping scheme is adopted. Introduce a time interval $\mathcal{I} = (0, T) \in \mathbb{R}^+$, and let $0 = t_0 \leq t_1 \leq \cdots \leq t_M$ be a partition of $\mathcal{I}$, with $t_{n+1} = t_n + \delta t$. The chosen time-stepping scheme is used to derive the time-step $n+1$ using information from the previous time-steps.

A variety of different schemes are available: explicit time integration schemes advance to the $n + 1$-th time-step using only information obtained from the n-th timestep or earlier, while implicit schemes require iterative methods as the information on the $n + 1$-th step also plays a role in the time-stepping procedure. Explicit integration schemes are favoured for their efficiency and ease of implementation, and perform best when employed for highly dynamic problems; on the other hand, implicit schemes have stronger stability properties, allowing the adoption of larger timesteps in quasi-static problems.

In this thesis, an implicit approach will be adopted, by using a backwards differentiation formula (BDF). The most intuitive BDF scheme is BDF1, also known as backward Euler method. It is obtained by substituting the time derivative of the velocity with the incremental ratio:

$$\frac{d\boldsymbol{v}}{dt} = \frac{\boldsymbol{v}^{n+1} - \boldsymbol{v}^n}{\delta t} \tag{2.63}$$

Since BDF1 only has an order of convergence of 1, a 2^{nd} order BDF (BDF2)

may be adopted instead, by substituting the derivative with:

$$\frac{d\boldsymbol{v}}{dt} = \frac{3\boldsymbol{v}^{n+1} - 4\boldsymbol{v}^{n} + \boldsymbol{v}^{n-1}}{2\delta t} \tag{2.64}$$

Since (2.64) requires both $\boldsymbol{v}^n$ and $\boldsymbol{v}^{n-1}$, BDF1 is used in the first iteration of the time-stepping procedure, when $n = 0$. In either case, the discretized time derivative will be substituted to the first term of the weak form (2.61).

2.5 Algebraic System

The FEM problem (2.61) is a linear system of equations. In algebraic form, it is expressed as

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{v}^{n+1} \\ \boldsymbol{p}^{n+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F} \\ \boldsymbol{0} \end{pmatrix} \tag{2.65}$$

Assuming $d = 2$ (the process would be analogous otherwise), $\boldsymbol{v}$ is a $2N_n$-dimensional vector containing both components of the velocity $\boldsymbol{v}_i$ on each node, and $\boldsymbol{p}$ is a N_n-dimensional vector containing the pressure p_i on each node:

$$\boldsymbol{v} = \left\{(v_1)_1, (v_2)_1, \ldots, (v_1)_{N_n}, (v_2)_{N_n}\right\} \tag{2.66}$$

$$\boldsymbol{p} = \{p_1, \ldots, p_{N_n}\} \tag{2.67}$$

The entries of the system matrix, which will be referred to as $\mathcal{A}$, are:

$$\boldsymbol{A} = \frac{\alpha\rho}{\delta t}\boldsymbol{M} + \boldsymbol{K} = \begin{pmatrix} \frac{\alpha\rho}{\delta t}\boldsymbol{M}_1 + \boldsymbol{K}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \frac{\alpha\rho}{\delta t}\boldsymbol{M}_2 + \boldsymbol{K}_2 \end{pmatrix} \tag{2.68}$$

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \end{pmatrix} \tag{2.69}$$

with $\alpha = 1$ when $n = 0$, and $\alpha = 3/2$ otherwise (see (2.63) and (2.64)). Notice that $\boldsymbol{A}$ is symmetric, and so too is the system matrix $\mathcal{A}$. The two matrices that add up to build $\boldsymbol{A}$ in (2.68) are the mass matrix $\boldsymbol{M}$ and stiffness matrix $\boldsymbol{K}$. For each element e these terms are, respectively:

$$(\boldsymbol{M}_{1,2})^e_{i,j} = \rho \int_{T_e} \phi_i \phi_j \, d\boldsymbol{x} \tag{2.70}$$

$$(\boldsymbol{K}_{1,2})^e_{i,j} = \mu \int_{T_e} \nabla\phi_i \cdot \nabla\phi_j \, d\boldsymbol{x} \tag{2.71}$$

$$(\boldsymbol{B}_{1,2})^e_{i,j} = \int_{T_e} \phi_i \partial_{x,y} \phi_j \; d\boldsymbol{x} \tag{2.72}$$

Similarly, the right-hand side vector is, element by element:

$$(\boldsymbol{F}_{1,2})^e_i = \begin{cases} \displaystyle\int_{T_e} (f_{1,2} + \frac{\rho}{\delta t} v^n_{1,2})_i \phi_i \; d\boldsymbol{x} & \text{if } n = 0 \\ \displaystyle\int_{T_e} (f_{1,2} + \frac{2\rho}{\delta t} v^{n+1}_{1,2} - \frac{\rho}{2\delta t} v^n_{1,2})_i \phi_i \; d\boldsymbol{x} & \text{otherwise} \end{cases} \tag{2.73}$$

where $(f_{1,2})_i$ are the components of the external forces, evaluated at node i, and where the known terms arising from the time discretization have been moved to the right-hand side. These local element contributions will have to be assembled to build the global system matrix and right-hand side vector. In the Finite Element method, all the integrals above are computed using a Gaussian quadrature, while in the Material Point Method, the material points themselves are used as integration weights, as will be discussed more in detail in 3.2.

2.6 Solution Scheme

An Updated Lagrangian formulation is adopted in this thesis; the equations are therefore solved for the increments $(\delta \boldsymbol{v}^{n+1}, \delta p^{n+1})$ rather than $(\boldsymbol{v}^{n+1}, p^{n+1})$:

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta \boldsymbol{v}^{n+1} \\ \delta \boldsymbol{p}^{n+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F} \\ \boldsymbol{0} \end{pmatrix} \tag{2.74}$$

These increments are obtained using an iterative scheme, as required by implicit time integration schemes; in particular, a predictor-corrector method is used. For each time-step, an initial guess of the incremental variables is first provided:

$$\delta \boldsymbol{v}^{n+1,0} = 0 \tag{2.75}$$

$$\delta p^{n+1,0} = 0 \tag{2.76}$$

Then, on the k-th iteration, new values $(\delta \boldsymbol{v}^{n+1,k}, \delta p^{n+1,k})$ are computed through the algebraic system (2.74) and are used to correct the previous guess:

$$\delta \boldsymbol{v}^{n+1,k} = \delta \boldsymbol{v}^{n+1,k} + \delta \boldsymbol{v}^{n+1,k-1} \tag{2.77}$$

$$\delta p^{n+1,k} = \delta p^{n+1,k} + \delta p^{n+1,k-1} \tag{2.78}$$

This process is iterated until convergence under a chosen criterion. At the end of each time-step, the kinematic quantities are then updated as:

$$\boldsymbol{v}_i^{n+1} = \delta \boldsymbol{v}_i^{n+1} + \boldsymbol{v}_i^n \tag{2.79}$$

$$p_i^{n+1} = \delta p_i^{n+1} + p_i^n \tag{2.80}$$

$$\boldsymbol{u}_i^{n+1} = \boldsymbol{u}_i^n + \delta t \boldsymbol{v}_i^{n+1} \tag{2.81}$$

2.7 Stabilization

In this section, the root causes of the numerical instability afflicting the discrete mixed formulation of the Stokes problem are explained, and a commonly used approach to amend them is presented: the Variational Multiscale (VMS) method.

The well-posedness of a weak formulation requires three conditions to be fulfilled:

- existence of the solution;
- uniqueness of the solution;
- the solution must be a continuous function of the data, that is, small variations δ of the data lead to small variations of the solution; in a more formal way,

  $$||\tilde{u} - u|| < C||\delta||, \tag{2.82}$$

  where $\tilde{u}$ and u are the perturbed and original solution respectively, C is a bounding constant, and $||\cdot||$ is an appropriate norm (in this case, the H^1 norm for velocity and the L^2 norm for pressure).

These conditions can be checked to be true in the continuous weak formulation (2.49) through Brezzi's splitting theorem which, not unlike the Lax Milgram theorem, imposes conditions on the bilinear forms.

Definition 2.7.1 (H^1_{div} space)**.** $H^1_{\text{div}}(\Omega)$ is the space of square integrable functions with square integrable divergence:

$$H^1_{\text{div}}(\varphi(\Omega)) = \left\{ \boldsymbol{v} \in L^2(\varphi(\Omega), \mathbb{R}^d) : \nabla \cdot \boldsymbol{v} \in L^2(\varphi(\Omega)) \right\} \tag{2.83}$$

□

Theorem 2.7.2 (Brezzi splitting theorem)**.** Let $\mathcal{V} = H^1_{\text{div}}(\varphi(\Omega))$, $\mathcal{Q} = L^2_0(\varphi(\Omega))$. Let $a : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$, $b : \mathcal{V} \times \mathcal{Q} \to \mathbb{R}$ and $f : \mathcal{V} \to \mathbb{R}$ be, respectively, the bilinear forms and the linear functional describing the weak formulation of the Stokes problem. Suppose that the following hypotesis hold true:

- a is continuous and coercive, that is:

$$\exists \alpha > 0 \text{ s.t.} \qquad |a(\boldsymbol{v}, \boldsymbol{v})| \geq \alpha ||\boldsymbol{v}||^2_{\mathcal{V}} \qquad \forall \boldsymbol{v} \in \mathcal{V} \tag{2.84}$$

- b is continuous and satisfies the Ladyzhenskaya–Babuška–Brezzi (LBB) or *inf-sup* condition (also known as "weak coercivity"):

$$\exists \beta > 0 \text{ s.t.} \qquad \inf_{q \in \mathcal{Q}} \sup_{\boldsymbol{v} \in \mathcal{V}} \frac{b(\boldsymbol{v}, q)}{||\boldsymbol{v}||_{\mathcal{V}} ||q||_{\mathcal{Q}}} \geq \beta \tag{2.85}$$

- f is continuous.

Then, the problem has a unique solution $(\boldsymbol{v}, p) \in \mathcal{V} \times \mathcal{Q}$ which statisfies the stability property:

$$||\boldsymbol{v}||_{\mathcal{V}} + ||p||_{\mathcal{Q}} \leq C ||f||_{\mathcal{V}^*} \tag{2.86}$$

□

Remark 2.7.3. The theorem requires $\boldsymbol{v} \in H^1_{\text{div}}$; however, since $H^1(\varphi(\Omega), \mathbb{R}^d) \subset H^1_{\text{div}}(\varphi(\Omega))$, the definition of $\mathcal{V}$ that has been used so far can remain unchanged.

To employ Brezzi's splitting theorem, the three conditions must be proven true. Continuity can be readily checked using the Cauchy-Schwartz (C.S.) inequality:

$$|a(\boldsymbol{v}, \boldsymbol{v})| \overset{\text{C.S.}}{\leq} \mu ||\boldsymbol{v}||_{L^2} ||\boldsymbol{v}||_{L^2} \leq \mu ||\boldsymbol{v}||_{\mathcal{V}} ||\boldsymbol{v}||_{\mathcal{V}} \tag{2.87}$$

$$|b(\boldsymbol{v}, q)| \overset{\text{C.S.}}{\leq} ||\boldsymbol{v}||_{L^2} ||q||_{\mathcal{Q}} \leq ||\boldsymbol{v}||_{\mathcal{V}} ||q||_{\mathcal{Q}} \tag{2.88}$$

$$|f(\boldsymbol{v})|| \overset{\text{C.S.}}{\leq} ||\boldsymbol{f}||_{L^2} ||\boldsymbol{v}||_{L^2} \tag{2.89}$$

Coercivity of a can be proven as follows. First, introduce the Poincaré inequality (proof will not be given):

$$||\boldsymbol{v}||_{L^2} \leq C_\Omega ||\nabla \boldsymbol{v}||_{L^2} \tag{2.90}$$

through which the following relation can be estabilished:

$$||\boldsymbol{v}||_{\mathcal{V}}^2 = ||\boldsymbol{v}||_{L^2}^2 + ||\nabla \boldsymbol{v}||_{L^2}^2 \leq (1 + C_\Omega^2)||\nabla \boldsymbol{v}||_{L^2}^2 \tag{2.91}$$

Then, coercivity can be shown:

$$a(\boldsymbol{v}, \boldsymbol{v}) = \mu||\nabla \boldsymbol{v}||_{L^2}^2 \geq \frac{\mu}{1 + C_\Omega^2}||\boldsymbol{v}||_{\mathcal{V}}^2 \tag{2.92}$$

The fulfillment of the *inf-sup* condition can be proven by introducing the following lemma (whose proof will be omitted):

Lemma 2.7.4. For every $p \in L_0^2(\varphi(\Omega))$ there exists $\boldsymbol{v}_p \in H_0^1(\varphi(\Omega))$ such that:

$$\nabla \cdot \boldsymbol{v}_p = p \qquad \text{and} \qquad ||\boldsymbol{v}||_{H^1} \leq C||p||_{L^2} \tag{2.93}$$

Then, notice that (2.85) can be reformulated as:

$$\sup_{\boldsymbol{v} \in \mathcal{V}} \frac{b(\boldsymbol{v}, q)}{||\boldsymbol{v}||_{\mathcal{V}}} \geq \beta||q||_{\mathcal{Q}} \qquad \forall q \in \mathcal{Q} \tag{2.94}$$

Substituting (2.93) into (2.94) yields:

$$\sup_{\boldsymbol{v} \in \mathcal{V}} \frac{b(\boldsymbol{v}, q)}{||\boldsymbol{v}||_{\mathcal{V}}} \geq \frac{b(\boldsymbol{v}_p, q)}{C||q||_{\mathcal{Q}}} = \frac{(q, q)}{C||q||_{\mathcal{Q}}} = \frac{1}{C}||q||_{\mathcal{Q}} \qquad \forall q \in \mathcal{Q} \tag{2.95}$$

By identifying $\beta = 1/C$, the *inf-sup* condition is verified.

So far, only the continuous weak problem has been proven to always be well-posed. This does not, however, necessarily translate into a well-posed discretized problem too; the discrete equivalent of theorem 2.7.2 is stated below.

Theorem 2.7.5. Let $a : \mathcal{V}_h \times \mathcal{V}_h \to \mathbb{R}$, $b : \mathcal{V}_h \times \mathcal{Q}_h \to \mathbb{R}$ and $f : \mathcal{V}_h \to \mathbb{R}$ be, respectively, the bilinear forms and the linear functional describing the discrete FEM formulation of the Stokes problem. If a is continuous and coercive:

$$\exists \alpha > 0 \text{ s.t.} \qquad |a(\boldsymbol{v}_h, \boldsymbol{w}_h)| \geq \alpha||\boldsymbol{v}||_{\mathcal{V}}^2 \tag{2.96}$$

and b is continuous and satisfies the discrete inf-sup condition:

$$\exists \beta > 0 \text{ s.t.} \qquad \inf_{q_h \in \mathcal{Q}_h} \sup_{\boldsymbol{v}_h \in \mathcal{V}_h} \frac{b(\boldsymbol{v}_h, q_h)}{||\boldsymbol{v}_h||_{\mathcal{V}} ||q_h||_{\mathcal{Q}}} \geq \beta \tag{2.97}$$

then the problem has a unique solution $(\boldsymbol{v}_h, p_h) \in \mathcal{V}_h \times \mathcal{Q}_h$ which satisfies the stability property:

$$||\boldsymbol{v}_h||_{\mathcal{V}} + ||p_h||_{\mathcal{Q}} \leq C||\boldsymbol{f}||_{\mathcal{V}^*} \tag{2.98}$$

□

In order for the conditions required by this theorem to be verified, a relation must hold between $\mathcal{V}_h$ and $\mathcal{Q}_h$, so that they may qualify as admissible FEM spaces. This relation is stated by the Fortin criterion.

Lemma 2.7.6 (Fortin criterion)**.** If the continuous inf-sup condition is satisfied, then the discrete inf-sup condition is satisfied if and only if there exists a linear operator $\Pi_h : \mathcal{V} \to \mathcal{V}_h$ such that:

$$b(\Pi_h \boldsymbol{v}, q) = b(\boldsymbol{v}, q) \qquad \forall q \in \mathcal{Q}_h \tag{2.99}$$

and

$$||\Pi_h \boldsymbol{v}_h||_{\mathcal{V}} \leq \gamma_h ||\boldsymbol{v}||_{\mathcal{V}} \qquad \forall \boldsymbol{v} \in \mathcal{V} \tag{2.100}$$

□

Remark 2.7.7. By linearity of the divergence and integral operators, (2.99) can alternatively be stated as:

$$\int_{\varphi(\Omega)} \nabla \cdot (\Pi_h \boldsymbol{v} - \boldsymbol{v}) q_h = 0 \qquad \forall \boldsymbol{v} \in H^1(\varphi(\Omega), \mathbb{R}^d), q_h \in \mathcal{Q}_h \tag{2.101}$$

This condition is guaranteed if

$$\nabla \cdot \mathcal{V}_h = \mathcal{Q}_h \tag{2.102}$$

which essentially means the velocity space has to be sufficiently "richer" than the pressure space.

The reason behind this requirement can be understood by identifying the system matrix $\mathcal{A}$, declared in (2.65), as that of a saddle point problem [41]. Indeed, one may introduce a Lagrangian functional $\mathfrak{L}(\boldsymbol{v}, \boldsymbol{p})$ whose stationary points coincide

with the solution $(\boldsymbol{v}^*, \boldsymbol{p}^*)$ of the linear system. In other words, this functional should be such that

$$\begin{cases} \nabla_{\boldsymbol{v}} \mathfrak{L}(\boldsymbol{v}, \boldsymbol{p}) \cdot \boldsymbol{w} = 0 \\ \nabla_p \mathfrak{L}(\boldsymbol{v}, \boldsymbol{p}) \cdot \boldsymbol{q} = 0 \end{cases} \quad \Leftrightarrow \quad \begin{cases} \boldsymbol{A}\boldsymbol{v} + \boldsymbol{B}^T \boldsymbol{p} = \boldsymbol{f} \\ \boldsymbol{B}\boldsymbol{v} = 0 \end{cases} \tag{2.103}$$

where $\boldsymbol{v} \in \mathbb{R}^n$ and $\boldsymbol{p} \in \mathbb{R}^m$ are the vectors of solutions.

This is satisfied by

$$\mathfrak{L}(\boldsymbol{v}, \boldsymbol{p}) = \frac{1}{2} \boldsymbol{v}^T \boldsymbol{A} \boldsymbol{v} - \boldsymbol{f}^T \boldsymbol{v} + (\boldsymbol{B}\boldsymbol{v})^T \boldsymbol{p}; \tag{2.104}$$

every solution $(\boldsymbol{v}^*, \boldsymbol{p}^*)$ is a saddle point of $\mathfrak{L}(\boldsymbol{v}, \boldsymbol{p})$, that is:

$$\mathfrak{L}(\boldsymbol{v}^*, \boldsymbol{p}) \leq \mathfrak{L}(\boldsymbol{v}^*, \boldsymbol{p}^*) \leq \mathfrak{L}(\boldsymbol{v}, \boldsymbol{p}^*) \qquad \forall \boldsymbol{v} \in \mathbb{R}^n, \quad \forall \boldsymbol{p} \in \mathbb{R}^m \tag{2.105}$$

or, alternatively:

$$\min_{\boldsymbol{v}} \max_{\boldsymbol{p}} \mathfrak{L}(\boldsymbol{v}, \boldsymbol{p}) = \mathfrak{L}(\boldsymbol{v}^*, \boldsymbol{p}^*) = \max_{\boldsymbol{p}} \min_{\boldsymbol{v}} \mathfrak{L}(\boldsymbol{v}, \boldsymbol{p}) \tag{2.106}$$

Identifying $\boldsymbol{p}$ as the vector of Lagrangian multipliers and $\boldsymbol{B}\boldsymbol{v}$ as the mass conservation constraint, the problem can be seen as a contrained minimization problem:

$$\begin{aligned} &\min_{\boldsymbol{v} \in \mathbb{R}^n} \frac{1}{2} \boldsymbol{v}^T \boldsymbol{A} \boldsymbol{v} - \boldsymbol{f}^T \boldsymbol{v} \\ &\text{subject to } \boldsymbol{B}\boldsymbol{v} = 0 \end{aligned} \tag{2.107}$$

It is expected that the number of constraints m must be smaller than the number of equations n, in order to have a well-defined problem; this justifies the relation between the solution spaces mentioned in (2.102).

On the algebraic system, well-posedness is given by Benzi's theorem:

Theorem 2.7.8 (Benzi)**.** Let $\boldsymbol{A}$ be a symmetric and semi-positive definite matrix, and let $\boldsymbol{B}$ be of max rank ($\text{rank}(\boldsymbol{B}) = n$). Then, matrix

$$\mathcal{A} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{pmatrix} \tag{2.108}$$

is invertible if and only if

$$\ker(\boldsymbol{A}) \cap \ker(\boldsymbol{B}) = \emptyset \tag{2.109}$$

□

Under the assumptions of this theorem, conditions (2.96) and (2.97) hold as follows:

$$\boldsymbol{v}^T \boldsymbol{A} \boldsymbol{v} \geq \alpha ||\boldsymbol{v}||^2 \tag{2.110}$$

$$\inf_{\boldsymbol{p} \in \mathbb{R}^m} \sup_{\boldsymbol{v} \in \mathbb{R}^n} \frac{\boldsymbol{p}^T \boldsymbol{B} \boldsymbol{v}}{||\boldsymbol{p}||\ ||\boldsymbol{v}||} \geq \beta \tag{2.111}$$

As mentioned when detailing the spatial discretization, linear basis functions are used for both velocity and pressure ($\mathcal{P}_1/\mathcal{P}_1$); this space can be shown to violate the inf-sup condition. If the basis functions are linear, then $\boldsymbol{v}_h$ is piecewise linear; in turn, $\nabla \cdot \boldsymbol{v}_h$ is piecewise constant in $T_e \in \mathcal{T}_h$. Moreover, consider a discrete pressure field with nodal values $(-1, 0, 1)$ at the three vertices of each triangular element $T_e \in \mathcal{T}_h$; this pressure field is, by construction, such that $\sum_i p_i = 0$ on each element. Then:

$$\begin{aligned} b(\boldsymbol{v}_h, p_h) &= \int_{\varphi(\Omega)} p_h \nabla \cdot \boldsymbol{v}_h \ d\boldsymbol{x} = \sum_{T_e \in \mathcal{T}_h} \nabla \cdot \boldsymbol{v}_h \int_{T_e} p_h \\ &= \sum_{T_e \in \mathcal{T}_h} \nabla \cdot \boldsymbol{v}_h \sum_{i=1}^{3} \frac{p_i |T_e|}{3} = 0; \end{aligned} \tag{2.112}$$

thus, there exists a non-trivial solution $(\boldsymbol{v}, p)$ such that (2.97) does not hold. Additionally, since there exists $p_h \neq 0$ such that $b(\boldsymbol{v}_h, p_h) = 0$, the null set of $\boldsymbol{B}$ is such that $\dim(\ker(\boldsymbol{B})) \neq 0$; therefore, Benzi's theorem cannot be applied, as it requires $\boldsymbol{B}$ to be full rank. The latter remark suggests that the instability arising from the violation of the inf-sup condition manifests in spurious oscillations on the pressure.

Inf-sup stability of the Stokes problem can be achieved by either choosing $\mathcal{P}_k/\mathcal{P}_{k-1}$ spaces with $k \geq 2$ (*Taylor-Hood* spaces), or applying an appropriate stabilization technique. The idea behind stabilization is to replace the lower right empty block in $\mathcal{A}$ with an invertible matrix $\boldsymbol{C}$, in order to make matrix $\mathcal{A}$ as a whole always invertible. Introducing $\boldsymbol{C}$ means committing a variational

crime, so this matrix should be such that

$$\lim_{h \to 0} \boldsymbol{C} = \boldsymbol{0} \tag{2.113}$$

2.7.1 Variational Multiscale Method

In this thesis, stabilization of the discretized weak form is achieved by employing the Variational Multiscale (VMS) method [33, 42, 43, 44]. The core idea behind this method lies in identifying and amending the inability of Finite Element methods to handle multiscale problems. Indeed, the solution of the continuous problem can be decomposed into coarse scales, which are solved by FEM, and fine subscales, which are neglected by numerical approximations. The VMS method attempts to describe the latter scales analytically, rather than numerically.

Let

$$\boldsymbol{V} = \boldsymbol{V}_h + \tilde{\boldsymbol{V}} \tag{2.114}$$

$$\boldsymbol{W} = \boldsymbol{W}_h + \tilde{\boldsymbol{W}} \tag{2.115}$$

be the sum decomposition of the continuous solution $\boldsymbol{V}$ and the test functions $\boldsymbol{W}$ into their coarse scales ($\boldsymbol{V}_h$ and $\boldsymbol{W}_h$ respectively) and their fine scales ($\tilde{\boldsymbol{V}}$ and $\tilde{\boldsymbol{W}}$ respectively).

The solution space $\mathcal{W}$ can similarly be expressed through the direct sum

$$\mathcal{W} = \mathcal{W}_h \oplus \tilde{\mathcal{W}} \tag{2.116}$$

where $\mathcal{W}_h$ is the finite element subspace and $\tilde{\mathcal{W}}$ is the infinite-dimensional fine scale subspace. The subscales are such that

$$\tilde{\boldsymbol{v}} = 0 \quad \text{on } \varphi(\partial\Omega_D) \quad \forall \tilde{\boldsymbol{v}} \in \tilde{\mathcal{V}} \tag{2.117}$$

$$\tilde{\boldsymbol{w}} = 0 \quad \text{on } \varphi(\partial\Omega_D) \quad \forall \tilde{\boldsymbol{w}} \in \tilde{\mathcal{V}} \tag{2.118}$$

Substituting (2.114) and (2.115) into the weak form (2.49) yields:

$$\begin{aligned} \left\langle \mathcal{D}_t(\boldsymbol{V}_h + \tilde{\boldsymbol{V}}), \boldsymbol{W}_h + \tilde{\boldsymbol{W}} \right\rangle_{\varphi(\Omega)} + B(\boldsymbol{V}_h + \tilde{\boldsymbol{V}}, \boldsymbol{W}_h + \tilde{\boldsymbol{W}}) \\ = F(\boldsymbol{W}_h + \tilde{\boldsymbol{W}}) \qquad \forall \boldsymbol{W}_h \in \mathcal{W}_h, \tilde{\boldsymbol{W}} \in \tilde{\mathcal{W}} \end{aligned} \tag{2.119}$$

The subscales can be assumed to be quasi-static, for simplicity. Due to the linear independence of $\boldsymbol{W}_h$ and $\tilde{\boldsymbol{W}}$, (2.119) can be split in two problems:

$$\langle \mathcal{D}_t(\boldsymbol{V}_h), \boldsymbol{W}_h \rangle_{\varphi(\Omega)} + B(\boldsymbol{V}_h, \boldsymbol{W}_h) + B(\tilde{\boldsymbol{V}}, \boldsymbol{W}_h) = F(\boldsymbol{W}_h) \quad \forall \boldsymbol{W}_h \in \mathcal{W}_h \tag{2.120}$$

$$\left\langle \mathcal{D}_t(\boldsymbol{V}_h), \tilde{\boldsymbol{W}} \right\rangle_{\varphi(\Omega)} + B(\boldsymbol{V}_h, \tilde{\boldsymbol{W}}) + B(\tilde{\boldsymbol{V}}, \tilde{\boldsymbol{W}}) = F(\tilde{\boldsymbol{W}}) \quad \forall \tilde{\boldsymbol{W}} \in \tilde{\mathcal{W}} \tag{2.121}$$

Problem (2.120) coincides with the discrete weak form (2.61), with an additional term representing the subscales. This term will be modeled using (2.121).

To this end, apply integration by parts to (2.121), obtaining

$$\begin{aligned} &\sum_e \left\langle \mathcal{D}_t(\boldsymbol{V}_h), \tilde{\boldsymbol{W}} \right\rangle_e + \sum_e \left\langle \mathcal{L}(\boldsymbol{V}_h), \tilde{\boldsymbol{W}} \right\rangle_e \\ &\quad + \sum_e \left\langle \mathcal{L}(\tilde{\boldsymbol{V}}), \tilde{\boldsymbol{W}} \right\rangle_e = \sum_e \left\langle \mathcal{F}, \tilde{\boldsymbol{W}} \right\rangle_e \quad \forall \tilde{\boldsymbol{W}} \in \tilde{\mathcal{W}}, \end{aligned} \tag{2.122}$$

where the notation introduced in (**??**) has been adopted; equation (2.122) is equivalent to applying an L^2 projection $\tilde{P}$ onto the space of $\tilde{\boldsymbol{W}}$:

$$\tilde{P}\left[\mathcal{L}(\tilde{\boldsymbol{V}})\right] = \tilde{P}\left[\mathcal{F} - \mathcal{D}_t(\boldsymbol{V}_h) - \mathcal{L}(\boldsymbol{V}_h)\right] \coloneqq \tilde{P}\left[\boldsymbol{\mathcal{R}}_h(\boldsymbol{V}_h)\right], \tag{2.123}$$

where $\boldsymbol{\mathcal{R}}_h(\boldsymbol{V}_h)$ is the residual.

The problem

$$\begin{cases} \tilde{P}\left[\mathcal{L}(\tilde{\boldsymbol{V}})\right] = \tilde{P}\left[\boldsymbol{\mathcal{R}}_h(\boldsymbol{V}_h)\right] & \text{in } \varphi(\Omega) \\ \tilde{\boldsymbol{V}} = 0 & \text{on } \varphi(\partial\Omega_D) \end{cases} \tag{2.124}$$

has a unique solution $\boldsymbol{V}_h \in \mathcal{W}_h$ which can be written as

$$\tilde{\boldsymbol{V}} = M_e(\boldsymbol{\mathcal{R}}_h(\boldsymbol{V}_h)) \tag{2.125}$$

where M_e is a bounded linear functional $M_e : H^{-1}(T_e) \to H_0^1(T_e)$. In order to simulate the subscales, M_e may be chosen as $\boldsymbol{\tau}_e \tilde{P}$, with $\boldsymbol{\tau}_e$ being the matrix of

stabilization parameters

$$\boldsymbol{\tau}_e = \begin{pmatrix} \boldsymbol{\tau}_{1,e} & 0 \\ 0 & \tau_{2,e} \end{pmatrix}, \qquad \boldsymbol{\tau}_{1,e} = \tau_{1,e}\boldsymbol{I}_d \tag{2.126}$$

Different stabilization methods will arise based on the choice of $\tilde{P}$. The Algebraic Sub-Grid Scales (ASGS) stabilization [45] will be used in this thesis, which imposes $\tilde{P} = \boldsymbol{I}$, such that

$$\tilde{\boldsymbol{V}} = \boldsymbol{\tau}_e \boldsymbol{\mathcal{R}}_h \tag{2.127}$$

Alternatively, the Orthogonal Sub-Grid Scales (OSGS) method may have been adopted [46], by considering $\tilde{P} = \boldsymbol{I} - P_h$ instead, where P_h is an L^2 projection onto the finite element space.

The stabilization parameters can be determined through a Fourier analysis of (2.124) [47]. Note that, in this analysis, the transient terms will be neglected for simplicity.

Definition 2.7.9 (Fourier transform)**.** Let $g(\boldsymbol{x}) \in L^1(\mathbb{R}^d)$. The function

$$\hat{g}(\boldsymbol{k}) \coloneqq \int_{\mathbb{R}^d} e^{-\mathrm{i}\frac{\boldsymbol{k}\cdot\boldsymbol{x}}{h}} g(\boldsymbol{x})\ d\boldsymbol{x} \tag{2.128}$$

with i being the imaginary unit and $\boldsymbol{k}$ being a dimensionless wave number. □

The Fourier transform of any function $g(\boldsymbol{x})$ defined on T_e is:

$$\hat{g}(\boldsymbol{k}) = \int_{T_e} e^{-\mathrm{i}\frac{\boldsymbol{k}\cdot\boldsymbol{x}}{h}} g(\boldsymbol{x})\ d\boldsymbol{x} \tag{2.129}$$

The Fourier transform of its derivative is:

$$\widehat{\frac{\partial \square}{\partial x_i}}(\boldsymbol{k}) = \int_{\partial T_e} n_i e^{\mathrm{i}\frac{\boldsymbol{k}\cdot\boldsymbol{x}}{h}} \square(\boldsymbol{x})\ ds + \mathrm{i}\frac{k_i}{h}\hat{\square}(\boldsymbol{k}); \tag{2.130}$$

however, the subscales are by definition dominated, in their Fourier representation, by the components with high wave number. The first term can thus always be neglected, leading to

$$\widehat{\frac{\partial \square}{\partial x_i}}(\boldsymbol{k}) \approx \mathrm{i}\frac{k_i}{h}\hat{\square}(\boldsymbol{k}) \tag{2.131}$$

Using this approximation, as well as the definition of $\mathcal{L}$ given in (2.36), (2.124)

can be developed as:

$$\begin{cases} \mu\dfrac{|\boldsymbol{k}^2|}{h^2}\hat{\tilde{\boldsymbol{v}}}(\boldsymbol{k}) + \mathrm{i}\dfrac{\boldsymbol{k}}{h}\hat{\tilde{p}}(\boldsymbol{k}) = \hat{\boldsymbol{\mathcal{R}}}_{h,1}(\boldsymbol{k}) & \text{(2.132a)} \\ \mathrm{i}\dfrac{\boldsymbol{k}}{h}\cdot\hat{\tilde{\boldsymbol{v}}}(\boldsymbol{k}) = \hat{\mathcal{R}}_{h,2}(\boldsymbol{k}) & \text{(2.132b)} \end{cases}$$

where $\boldsymbol{\mathcal{R}}_{h,1}$ and $\mathcal{R}_{h,2}$ are the two components of $\boldsymbol{\mathcal{R}}_h$; the former may be assumed to be divergence-free. Multiplying (2.132a) by $\mathrm{i}\boldsymbol{k}/h$ and exploiting the divergence-free nature of $\boldsymbol{\mathcal{R}}_{h,1}$, then substituting (2.132b) into (2.132a), yields:

$$\mu\frac{|\boldsymbol{k}|^2}{h^2}\hat{\mathcal{R}}_{h,2}(\boldsymbol{k}) - \frac{|\boldsymbol{k}^2|}{h^2}\hat{\tilde{p}}(\boldsymbol{k}) = 0 \tag{2.133}$$

The goal is to obtain an equation of the form $\hat{p}(\boldsymbol{x}) \approx \tau_{2,e}\mathcal{R}_{h,2}(\boldsymbol{x})$. To this end, rewrite (2.133) as

$$\hat{\tilde{p}}(\boldsymbol{k}) = \mu\hat{\mathcal{R}}_{h,2}(\boldsymbol{k}) \tag{2.134}$$

which easily translates into the aforementioned equation, by identifying

$$\tau_{2,e} = \mu \tag{2.135}$$

This result can be substituted in (2.132a) in order to obtain $\tilde{\boldsymbol{v}}(\boldsymbol{x}) = \tau_{1,e}\boldsymbol{\mathcal{R}}_{h,1}(\boldsymbol{x})$. Starting from

$$\hat{\tilde{\boldsymbol{v}}}(\boldsymbol{k}) = -\mathrm{i}\frac{h}{|\boldsymbol{k}^2|}\boldsymbol{k}\hat{\mathcal{R}}_{h,2}(\boldsymbol{k}) + \frac{h^2}{\mu|\boldsymbol{k}^2|}\hat{\boldsymbol{\mathcal{R}}}_{h,1}(\boldsymbol{k}), \tag{2.136}$$

the first term may be neglected; physically, this can be seen as neglecting the contribution in the subscales given by any error in satisfying the incompressibility constraint due to the finite element approximation. The remaining equation is

$$\hat{\tilde{\boldsymbol{v}}}(\boldsymbol{k}) = \frac{h^2}{\mu|\boldsymbol{k}^2|}\hat{\boldsymbol{\mathcal{R}}}_{h,1}(\boldsymbol{k}) \coloneqq \mathcal{T}(\boldsymbol{k})\hat{\boldsymbol{\mathcal{R}}}_{h,1} \tag{2.137}$$

In order to obtain $\tau_{1,e}$, introduce the Plancherel theorem.

Theorem 2.7.10 (Plancherel)**.** Let $g(\boldsymbol{x}) \in L^2(\mathbb{R}^d)$ and let $\hat{g}(\boldsymbol{k})$ be it Fourier Transform. The Plancherel theorem states that:

$$\int_{\mathbb{R}^d} |g(\boldsymbol{x})|^2 \, d\boldsymbol{x} = \int_{\mathbb{R}^d} |\hat{g}(\boldsymbol{k})|^2 \, d\boldsymbol{k} \tag{2.138}$$

□

Plancherel theorem is applied to (2.137):

$$||\tilde{\boldsymbol{v}}||^2_{L^2(T_e)} \approx \frac{1}{(2\pi)^d}||\hat{\tilde{\boldsymbol{v}}}||^2_{L^2(\mathbb{R}^d)} \approx \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} |\mathcal{T}(\boldsymbol{k})|^2 |\hat{\boldsymbol{\mathcal{R}}}_{h,1}(\boldsymbol{k})|\ d\boldsymbol{k} \tag{2.139}$$

The mean value theorem states that there exists a wave number $\boldsymbol{k}_0$ such that

$$\frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} |\mathcal{T}(\boldsymbol{k})|^2 |\hat{\boldsymbol{\mathcal{R}}}_{h,1}(\boldsymbol{k})|\ d\boldsymbol{k} = \frac{1}{(2\pi)^d}|\mathcal{T}(\boldsymbol{k}_0)|^2\int_{\mathbb{R}^d} |\hat{\boldsymbol{\mathcal{R}}}_{h,1}(\boldsymbol{k})|\ d\boldsymbol{k} \tag{2.140}$$

It is now easy to obtain

$$||\tilde{\boldsymbol{v}}(\boldsymbol{x})||_{T_e} \approx |\mathcal{T}(\boldsymbol{k}_0)|\ ||\boldsymbol{\mathcal{R}}_{h,1}(\boldsymbol{x})||_{T_e}, \tag{2.141}$$

through which it can be concluded that there exists a coefficient c such that

$$\tau_{1,e} = c\frac{h^2}{\mu} \tag{2.142}$$

Now that the subscales have successfully been expressed as (2.127), integration by parts is applied to $B(\tilde{\boldsymbol{V}}, \boldsymbol{W}_h)$ in order to single out $\tilde{\boldsymbol{V}}$:

$$B(\tilde{\boldsymbol{V}}, \boldsymbol{W}_h) = \sum_e \left\langle \mathcal{L}(\tilde{\boldsymbol{V}}), \boldsymbol{W}_h \right\rangle_e = \sum_e \left\langle \tilde{\boldsymbol{V}}, \mathcal{L}^*(\boldsymbol{W}_h) \right\rangle_e \tag{2.143}$$

with $\mathcal{L}^*$ being the adjoint of $\mathcal{L}$. To define $\mathcal{L}^*(\boldsymbol{W}_h)$, start from (2.41) and apply the divergence theorem to each term:

$$(-\mu\Delta\boldsymbol{v}, \boldsymbol{w})_{\mathbb{R}^d} = (\mu\nabla\boldsymbol{v}, \nabla\boldsymbol{w})_{\mathbb{R}^{d\times d}} = (-\mu\boldsymbol{v}, \Delta\boldsymbol{w})_{\mathbb{R}^d} \tag{2.144}$$

$$(\boldsymbol{w}, \nabla p)_{\mathbb{R}^d} = (-\nabla\cdot\boldsymbol{w}, p)_{\mathbb{R}} \tag{2.145}$$

$$(-\nabla\cdot\boldsymbol{v}, q)_{\mathbb{R}} = (-\boldsymbol{v}, \nabla q)_{\mathbb{R}^d} \tag{2.146}$$

This translates into the following adjoint operator:

$$\mathcal{L}^*(\boldsymbol{W}_h) = \begin{pmatrix} -\mu\Delta\boldsymbol{w}_h + \nabla q_h \\ -\nabla\cdot\boldsymbol{w}_h \end{pmatrix} \tag{2.147}$$

Note that, due to the linear nature of the discretized variables, $-\mu\Delta\boldsymbol{w}_h$ vanishes from (2.147). Now, substituting into (2.120), and taking into account the definition of $\boldsymbol{\mathcal{R}}_h$, the stabilized weak form becomes:

$$\begin{aligned}&\langle\mathcal{D}_t(\boldsymbol{V}_h),\boldsymbol{W}_h\rangle_{\varphi(\Omega)}+B(\boldsymbol{V}_h,\boldsymbol{W}_h)+\sum_e\langle-\boldsymbol{\tau_e}\mathcal{L}(\boldsymbol{V}_h),\mathcal{L}^*(\boldsymbol{W}_h)\rangle_e\\ =&F(\boldsymbol{W}_h)+\sum_e\langle-\boldsymbol{\tau_e}\mathcal{F},\mathcal{L}^*(\boldsymbol{W}_h)\rangle_e\qquad\forall\boldsymbol{W}_h\in\mathcal{W}_h\end{aligned}\tag{2.148}$$

Remark 2.7.11. Equation (2.148) differs from the unstabilized weak form (2.61) by two terms which have appeared in the right-hand side and left-hand side respectively.

The right-hand side contribution to can be developed, element by element, as:

$$\langle-\boldsymbol{\tau_e}\mathcal{F},\mathcal{L}^*(\boldsymbol{W}_h)\rangle_e=\langle-\tau_{1,e}\boldsymbol{f},\nabla q\rangle_e\tag{2.149}$$

Similarly, the left-hand side is:

$$\langle-\boldsymbol{\tau_e}\mathcal{L}(\boldsymbol{V}_h),\mathcal{L}^*(\boldsymbol{W}_h)\rangle_e=\langle-\tau_{1,e}\nabla p_h,\nabla q_h\rangle_e+\langle-\tau_{2,e}\nabla\cdot\boldsymbol{v}_h,\nabla\cdot\boldsymbol{w}_h\rangle_e\tag{2.150}$$

These contributions are then added to (2.74), to obtain the algebraic system describing the stabilized problem.

Stabilized mixed formulation of the Stokes problem:

$$\begin{pmatrix}\boldsymbol{A}+\boldsymbol{S}^v & \boldsymbol{B}^T\\ \boldsymbol{B} & \boldsymbol{S}^p\end{pmatrix}\begin{pmatrix}\delta\boldsymbol{v}^{n+1}\\ \boldsymbol{p}^{n+1}\end{pmatrix}=\begin{pmatrix}\boldsymbol{F}\\ \boldsymbol{R}^p\end{pmatrix}\tag{2.151}$$

The entries of the stabilization matrix $\boldsymbol{S}^v$ are:

$$\boldsymbol{S}^v=\begin{pmatrix}\boldsymbol{S}^v_{1,1} & \boldsymbol{S}^v_{1,2}\\ \boldsymbol{S}^v_{2,1} & \boldsymbol{S}^v_{2,2}\end{pmatrix}\tag{2.152}$$

Each of these stabilization terms are defined, element by element, as follows:

$$\left(\boldsymbol{S}^v_{1,1}\right)^e_{i,j}=-\tau_2\int_{T_e}\partial_x\phi_i\partial_x\phi_j\;d\boldsymbol{x}\tag{2.153}$$

$$\left(\boldsymbol{S}^v_{1,2}\right)^e_{i,j} = -\tau_2 \int_{T_e} \partial_x \phi_i \partial_y \phi_j \; d\boldsymbol{x} \tag{2.154}$$

$$\left(\boldsymbol{S}^v_{2,1}\right)^e_{i,j} = -\tau_2 \int_{T_e} \partial_y \phi_i \partial_x \phi_j \; d\boldsymbol{x} \tag{2.155}$$

$$\left(\boldsymbol{S}^v_{2,2}\right)^e_{i,j} = -\tau_2 \int_{T_e} \partial_y \phi_i \partial_y \phi_j \; d\boldsymbol{x} \tag{2.156}$$

$$\left(\boldsymbol{S}^p\right)^e_{i,j} = -\tau_1 \int_{T_e} \nabla \phi_i \cdot \nabla \phi_j \; d\boldsymbol{x} \tag{2.157}$$

$$\left(\boldsymbol{R}^p\right)^e_{i,j} = -\tau_1 \int_{T_e} \boldsymbol{f} \cdot \nabla \phi \; d\boldsymbol{x} \tag{2.158}$$

Notice how, since the condition (2.113) for an admissible variational crime is satisfied by (2.157), this is an admissible stabilization.

Equation (2.151) is the stabilized, mixed velocity-pressure finite element formulation for the Stokes problem. Some additional steps are required to obtain the MPM mixed formulation, which will be described in 3.6.2.

Chapter 3

The Material Point Method

3.1 Structure

Inheriting its structure from FLIP, the Material Point Method adopts an hybrid structure, by using Lagrangian particles that flow through a periodically resetting Eulerian grid. The MPM does, however, diverge from FLIP in two core aspects. Unlike its predecessor, it approximates the problem using a finite element scheme, rather than adopting a finite difference approach, by iterating upon a FEM-based weak formulation of FLIP [48]. Unlike FLIP, MPM also stores stresses and strains as nodal information, as it was originally intended to be used in solid mechanics.

In the MPM, the body is discretized into N_p particles (or material points), which track all the information relevant to the motion, as previously described in Section 1.2.1. In the original MPM formulation, material points did not have a subdomain assigned to them, and the information was considered to be entirely localized within zero-dimensional particles whose shape is not tracked (but which have a mass and volume assigned to them); further developments of this method, such as GIMP [49] and CPDI [50], introduced deformable subregions to be assigned to the particles and tracked as well, in order to address the so-called "cell-crossing error"; a more in-depth description of this phenomenon will be provided in Section 3.4.2.

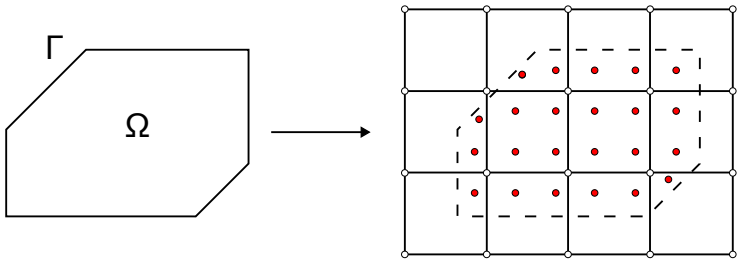


Figure 3.1: Discretization of a continuum Ω into material points over a computational grid, as shown in Zhang et al. [5].

On the other hand, the computational domain is discretized into an Eulerian background grid through N_n nodes. When the problem is solved on this mesh, either an Updated or a Total Lagrangian scheme may be adopted; the former choice is the most popular, and was adopted by the original MPM formulation, while the latter was introduced by de Vacourbeil et al. [51]. If an Updated Lagrangian formulation is used, then the grid must cover the space occupied by the body during the whole transformation, while in the case of a Total Lagrangian scheme, the mesh is only required to cover the space occupied by the body in its material configuration. The Updated Lagrangian scheme will be adopted in this thesis. Many MPM implementations opt for a uniform Cartesian grid, as neighbourhood searches to define connectivity are not required in this case, lightening the computational burden; basis functions are also easier to formulate in such grids. Unstructured grids are, however, the superior choice when complex boundary conditions have to be imposed, such as on curved surfaces, and are thust most commonly employed in a geomechanics setting [52, 53, 54, 55]. The latter approach will thus be adopted in this work.

The ability of the MPM to efficiently process large deformation problems lies in its hybrid Lagrangian-Eulerian structure: the material points provide a Lagrangian description of the continuum, while they flow through the Eulerian background grid. It is in this grid that the continuous fields describing the body's status (eg. displacement, velocity, pressure) are discretized, and it is here that the momentum balance equation is solved. Each node of the mesh is supplied with a basis function, used to project the data to and from the nodes, and to discretize the continuous fields. Note, however, that the mesh is only used to perform computations; any information held in the nodes is only of temporary use, and can be discarded after being mapped to the particles. The

mesh can thus be reconstructed at each timestep, circumventing the issue of excessive distortions afflicting purely Lagrangian schemes.

This structure allows the MPM to reap the benefits of both frameworks: on one hand, a Lagrangian discretization of the continuum is significantly more apt at describing large deformations and displacements; tracking the body's surfaces, which can be useful when applying boundary conditions on them, is also easier. On the other hand, the Eulerian background grid will never observe severe distortions, being independent from the body whose particles flow through.

Due to the discrete nature of the mass, which is fixed and stored in the particles, mass conservation is satisfied in MPM.

3.2 Algorithm

All MPM formulations follow the same general algorithm [7, 8], which dictates the interactions between the material points and the grid.

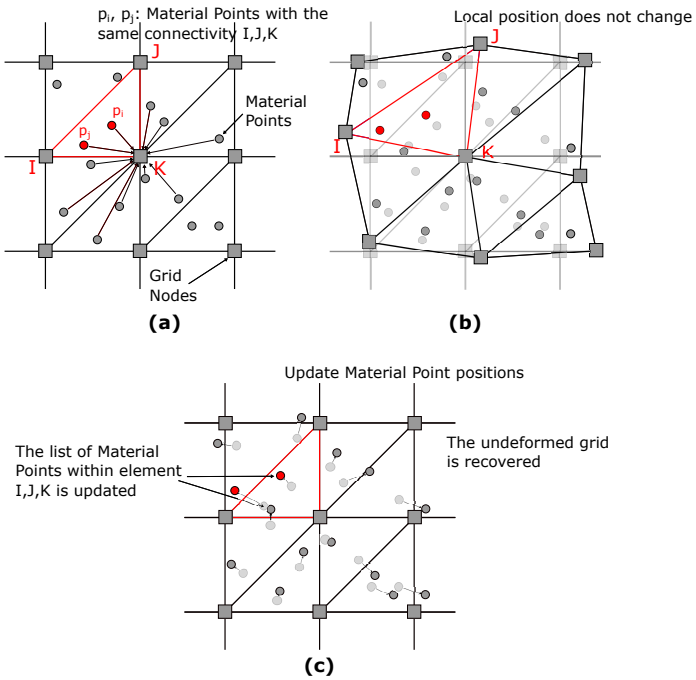


Figure 3.2: MPM phases, as shown in Iaconeta et al. [56]: (a) Initialization phase, (b) UL-FEM phase, (c) Convective phase

Initialization phase. MPM solves the problem on the background grid, but the information is initially stored in the particles. Thus, at the beginning of each timestep, the initialization phase is performed. First, the connectivity between material points is defined; then, the information stored in the particles is extrapolated onto the mesh nodes. This information pertains to the previous timestep t^n.

In the initialization phase, the nodal mass is first obtained by projecting the mass stored in each particle to the nodes, according to the previously defined connectivity:

$$m_i = \sum_{p=1}^{N_p} \phi_i(\boldsymbol{x}_p) m_p. \tag{3.1}$$

Then, the internal variables at the grid nodes can be interpolated from the material points p:

$$\square_i = \frac{1}{m_i} \sum_{p=1}^{N_p} m_p \phi_i(\boldsymbol{x}_p) \square_p. \tag{3.2}$$

UL-FEM phase. In the Updated Lagrangian FEM phase, the momentum balance equation is solved on the background grid, following a classical Finite Element procedure. The benefits of using a Lagrangian frame of reference manifest here: difficult to handle, nonlinear convective terms do not appear, unlike in Eulerian methods. Note that the material points are kept fixed in this phase (while the nodes are allowed to move) and, in fact, they act as integration points for the integrals of the weak formulation of the problem, while the volumes they store act as the corresponding integration weights. This is a crucial difference which distinguishes this method from FEM, where the integration points are always assumed to be Gaussian. For this reason, MPM tends to have suboptimal convergence, and overall perform worse than its mesh-based counterpart when it's applied to small displacement problems. In particular, each integral is approximated as

$$\int_\Omega f \, d\Omega = \sum_{p=1}^{N_p} f(\boldsymbol{x}_p) V_p, \tag{3.3}$$

with p being the integration points and V_p being the integration weights.

Through this procedure, the local components of the left-hand side matrix, as well as of the right-hand side vector, are defined and assembled into their global counterparts. The resulting system is solved in accordance to the chosen solution

scheme, using a standard Finite Element procedure.

Convective phase. Lastly, in the convective phase, the newly computed internal variables at time t^{n+1} are interpolated back to the material points, and their position is updated:

$$\square_p = \sum_{i=1}^{N_n} N_i(\boldsymbol{x}_p)\square_i, \tag{3.4}$$

with i being the related mesh nodes.

At the end of this process, the grid is reset. Since the material points are fixed in place when the grid is rebuilt, this defines a motion of the grid nodes relative to the particles. It is this motion that models convection in this algorithm, and this phase owes its name to this.

3.3 Numerical Aspects

3.3.1 Time Integration Schemes

As detailed in Section 2.4, a time integration scheme needs to be introduced in order to discretize and solve the time derivative that arises in the weak formulation. In this section, more information on the time integration schemes used within the context of MPM will be provided.

Explicit Methods

The original MPM formulation detailed by Sulsky adopted an explicit formulation. This approach is the most efficient one to solve fast, transient problems, which that paper focused on. Generally speaking, most available MPM algorithms adopt an explicit time integration scheme, as the resulting formulation is usually easier to conceptualize and deal with: explicit schemes only require knowledge of the variables at the current timestep t^n in order to predict their values at the next timestep $t^{n+1} = t^n + \delta t$, which allows for a much simpler algorithm to be adopted.

On the other hand, explicit approaches are known to be only conditionally stable, and as such they require small time increments δt in order to obtain a stable solution. This problem is negligible in fast transient problems, where

small time steps would be required anyway.

Implicit Methods

Fewer implicit formulations have been developed for MPM compared to their explicit counterparts. The first implicit formulations were developed by Cummins and Brackbill [57] for PIC, and by Guilkey and Weiss [58] for MPM. Implicit schemes are suitable when dealing with quasi-static problems, or problems with low loading rates. In this case, the conditional stability of explicit methods would force unreasonably short time steps in order to guarantee stability; the benefits of an implicit scheme's unconditional stability thus outweigh the additional computational effort they inherently require.

3.3.2 Formulations

As mentioned in Section 3.4.2, the original MPM formulation suffers from cell-crossing error, which results in spurious stress oscillations as particles cross over cell boundaries. Grid basis functions can be chosen such that their gradient is smooth across cells, thus amending this issue. A variety of different formulations have been developed for this purpose.

Characteristic functions

Characteristic functions have been implemented in numerous formulations in order to address cell-crossing errors [49]. Each material point is assigned to its own domain, each represented by a characteristic function $\chi_p(\boldsymbol{x})$.

Admissible characteristic functions must be a partition of unity in the initial configuration:

$$\sum_p \chi_p(\boldsymbol{x}) = 1 \quad \forall \boldsymbol{x} \tag{3.5}$$

Properties of the material points may be expressed through these characteristic functions:

$$V_p = \int_\Omega \chi_p(\boldsymbol{x}) \; d\boldsymbol{x} \tag{3.6}$$

$$m_p = \int_\Omega \rho(\boldsymbol{x}) \chi_p(\boldsymbol{x}) \; d\boldsymbol{x} \tag{3.7}$$

Remark 3.3.1. In the original MPM formulation, the particles had no spatial

domain assigned to them; in this case, the characteristic functions are implicitly set as the dirac delta:

$$\chi_p = \delta(\boldsymbol{x} - \boldsymbol{x}_p)V_p \tag{3.8}$$

Advanced MPM formulations implement weighting functions, defined as the convolution of the grid basis functions with the characteristic functions:

$$\phi_i(\boldsymbol{x}) = \frac{1}{V_p}\int_\Omega \chi_p(\boldsymbol{x})N_i(\boldsymbol{x})\ d\boldsymbol{x} \tag{3.9}$$

where N_i are the grid basis functions. Note that, in previous equations, grid basis functions had been referred to as ϕ_i; such notation was chosen to keep those formulations more general since, lacking specific indications, (3.8) is assumed, leading to $\phi_i \equiv N_i$.

Replacing the grid basis functions with appropriately chosen weighting functions can mitigate the cell-crossing error, as the discontinuity in the gradient is removed, while still allowing the use of linear basis functions. The most relevant attempts recorded in the literature are the Generalized Interpolation Material Point (GIMP) method [49], and the Convected Particle Domain Interpolation (CPDI) [50, 59].

Generalized Interpolation Material Point (GIMP). Gimp was first developed by Bardenhagen et al. in 2004 [49] and features piecewise-constant particle characteristic functions, commonly called top-hat functions:

$$\chi_p(\boldsymbol{x}) = \begin{cases} 1 & \text{when } \boldsymbol{x} \in \Omega_p \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

In 1D, the particle domains are usually given by segments; in 2D and 3D, they are described by rectangles and cuboids respectively, generated through the tensorial product of 1D domains. This approach allows a convenient analyical description of the domains.

Adopting such characteristic functions results in the following weighting functions:

$$\phi_i = \frac{1}{V_p}\int_{\varphi(\Omega_p)} N_i(\boldsymbol{x})d\Omega \tag{3.11}$$

Since the domain Ω_p evolves over time, so do the weighting functions; alterna-

tive GIMP formulations simplify this: namely, the unchanged GIMP (uGIMP) and the contiguous particle GIMP (cpGIMP) [60]. uGIMP keeps the particle domains unchanged by integrating over the reference configuration Ω_p instead, while cpGIMP takes into account deformations but neglects shear deformations, thus still allowing the use of analytical expressions to describe the domains.

In either case, having to adopt rectangular particle domain is a source of error in GIMP [50]. The problems that arise are twofold: first, this choice makes it impossible to fully cover complex computational domains; moreover, even when it is possible, this property would be lost after the body's deformation. This issue, which leads to gaps that prevent a complete description of the domain, is especially prominent when shear deformations or large rotations are involved. In this case, evaluating the weighting functions and their gradients may become very computationally taxing, as the integration domain would be split across multiple cells.

Convected Particle Domain Interpolation (CPDI). CPDI was developed by Sadeghirad et al. [50] as an improvement upon GIMP, in order to overcome its aforementioned limitations. It does so by describing the particle domains as parallelograms instead; an analytical description of such domains is made possible by introducing an alternative set of basis functions Q, which are defined within the parallelogram and which interpolate the grid basis functions exactly at its four vertices. The grid basis functions are thus approximated as:

$$N_i^{\text{app}} = \sum_{\alpha=1}^{4} Q_\alpha^p(\boldsymbol{x}) N_i(\boldsymbol{x}) \tag{3.12}$$

where Q_α^p is the alternative basis function of the α-th corner of the domain of particle p. Introducing this notation allows exact evaluation of integral (3.11).

Note that, since the particles are described as parallelograms, some gaps in the continuum description will still be present as deformation ensues; an improvement upon CPDI, called CPDI2, was later developed in Sadeghirad et al. [59]. Particle domains are described as quadrilaterals instead, such that the body can be tiled without gaps, even under arbitrary deformations. However, this approach is not exempt of drawbacks: the continuum must, effectively, be meshed in order to avoid gaps, which nullifies one of the main draws of MPM, that being the meshless nature of its material points, thus carrying the risk of

excessive deformation of the particle domains [61]; moreover, parallelization may be less immediate since particle domains may be such that they span multiple CPU domains [62].

Enriched basis functions

An alternative way to address cell crossing error is to enrich or increase the order of the basis functions; the goal is to remove the discontinuity of their derivative across cell boundaries.

Higher order basis functions. Higher-order functions innately have smooth gradients across cell boundaries, and also carry the benefit of improved convergence (an especially pressing issue in MPM, as mentioned previously), but their increased complexity will lead to worse computational times. Quadratic Bernstein functions [51] and both quadratic and cubic B-splines [63, 64] have been successfully been implemented in MPM.

Dual Domain MPM. Dual Domain MPM (DDMPM) Was first proposed by Zhang et al. [65] to mitigate cell-crossing errors. This method keeps the original, linear basis functions, without introducing characteristic functions, but instead modifies their gradient to remove their discontinuities across cell boundaries. In particular, the support of the gradient is enlarged, and is greater than that of the basis functions, thus increasing the influence domains of nodes and material points for quantities related to the gradient, while keeping the influence domain unchanged for other quantities.

3.4 Strengths, Limitations and Coupling

The following paragraphs will be structured as follows: first, the merits of the Material Point Method will be discussed, highlighting both its benefits and its drawbacks; then, it will be compared to the numerical methods introduced earlier in this thesis, in order to showcase why one may want to adopt the MPM to solve a given problem.

3.4.1 Advantages

Hybrid structure. The Material Point Method is a hybrid method, combining the main qualities of both mesh-based and mesh-free method. As discussed

in Section 1.2.1, this structure allows the MPM to bypass the mesh entanglement issues associated with purely Lagrangian methods, while still carrying the benefits of Lagrangian formulations when it comes to handling convection terms.

Flexible constitutive laws. MPM has been greatly appreciated in the field of geomechanics due to its ability to effortlessly handle complex constitutive laws; storing the constitutive parameters in each material point also makes it a good candidate for multi-phase problems [66], as different phases can be identified through different constitutive laws.

Contact detection. One of the most appreciated properties of the MPM, which sets it apart from other hybrid methods, is its proficiency at handling no-slip, no-penetration contact problems natively, without requiring additional contact detection algorithm [28]. These problems are easily handled by MPM due to the particle velocity field used to update the particle positions being single-valued and common for all particles, which inherently precludes interpenetration. Frictional contact between bodies still requires additional numerical treatments; additional contact algorithms have been developed to address this [67].

Similarities with FEM. The MPM scheme features several similarities with the Finite Element Method; in fact, it may be described as a Finite Element scheme where the integration points are allowed to move over time [58]. This similarity greatly enhances the amount of literature available for it, as preexisting FEM algorithms can be adapted to MPM as well. Sharing the same structure as FEM, it is also exceedingly easy to implement the Material Point Method within a parallel computing framework, compared to most mesh-free methods.

3.4.2 Shortcomings

Lower accuracy. Despite showing clear advantages over other methods when large displacement continuum mechanics problems are treated, the MPM struggles to be competitive with FEM in other cases, due to its inherently lower accuracy. Indeed, the material points act as moving integration points in the MPM scheme; FEM, on the other hand, adopts fixed Gauss points to approximate the integrals. Unlike Gauss Points, the particles in MPM may not be

located such as to guarantee optimal results. To mitigate this issue, a common strategy is to locate the material points on the Gauss points at the start of the simulation.

Formal analysis of the properties of MPM (such as convergence and stability) is also quite challenging due to the volatile nature of the integration points. Attempts to quantify quadrature and interpolation errors include [68] and [63], while mitigating techniques have been proposed in [49], [50] and [64].

Memory footprint. This downside only afflicts Updated Lagrangian formulation; in this case, the background grid has to cover the entire region the body will occupy over the simulation time. On the other hand, Total Lagrangian formulations only require the computational grid to cover the body's reference configuration [51].

Cell crossing instability. The original MPM formulation suffered from cell crossing instability. This error may arise when a particle crosses the boundary from one cell to another. If the basis functions are not smooth across cell boundaries, their gradients will exhibit a discontinuity across cells. Quantities that are related to the gradient of the basis functions (i.e. the internal forces) may thus experience imbalances right after a particle has crossed over a cell boundary. To understand this phenomenon, one may imagine a simple 1D case in solid mechanics, where a truss in quasi-static loading condition is discretized in two elements (each of width Δx) with two particles in each. Assume, then, a constant stress σ_0. The entries of the internal forces vector, in its weak form, is the following (a detailed analysis will not be provided, since the procedure to obtain a weak formulation is standard and was showcased previously):

$$f_i^{\text{int}} = -\int_\Omega \sigma_0 \partial_x \phi_i(x)\ d\Omega \approx -\sum_p \sigma_o \partial_x \phi_i(x_p) V_p \tag{3.13}$$

In the aforementioned example, the derivatives of the grid basis functions have constant values $\pm 1/\Delta x$ respectively on either element. Predictably (due to the quasi-static loading condition), the internal forces will be zero:

$$f^{int} = -\sigma_p V_p \left(\frac{1}{\Delta x} + \frac{1}{\Delta x} - \frac{1}{\Delta x} - \frac{1}{\Delta x} \right) = 0 \tag{3.14}$$

Right after cell crossing happens, however, this balance is disrupted and a nonzero value of the internal force unphysically arises. Various techniques have been developed to address this issue, which will be briefly discussed in Section 3.3.2.

Mapping errors. Inaccurate transfer of information from the background grid to the particles leads to momentum and energy conservation errors, as well as velocity projection error. As mentioned in Section 1.2.1, two possible schemes exist to transfer information: the PIC transfer and the FLIP scheme. PIC schemes are known to be more stable, but they also bring along significant energy dissipation; FLIP schemes, on the other hand, sacrifice accuracy in exchange for better energy conservation properties [22, 23]. Modern solutions to improve transfer quality include the affine PIC (APIC) [22] and Polynomial PIC (PolyPIC) [23], which augment the particles with locally affine and polynomial descriptions of the velocity, respectively; the Taylor PIC (TPIC) [69], which uses first-order Taylor series approximations for the velocity projection; and the hybrid PIC/FLIP, which combines both transfer schemes, as shown, for example, in [70, 71, 72, 73].

3.4.3 Comparison with Other Numerical Methods

Having laid out the main characteristics of the MPM, it is now possible to compare it with other numerical methods, to understand its use cases.

Most currently available commercial software employ the SPH method to solve large deformation problems. In an explicit framework, comparative studies between MPM and SPH, applied to hypervelocity impact problems, have been performed by Ma et al. [74]; in these tests, the MPM consistently outperformed its mesh-free counterpart, both in terms of velocity and accuracy. While the velocity increase can be partly attributed to the larger timesteps that the MPM allows to adopt, it should also be noted that the lack of a neighbour search algorithm also played a role; using such algorithms can only be avoided when employing Cartesian structured grids, however. The increased accuracy can be attributed to the lack of tensile instability that afflicts SPH, but also to the greater consistency of the shape functions in MPM.

In the context of fluid dynamics, the work of Sun et al. [75] compared the performance of MPM with SPH; despite the latter being the most prominent method

in CFD, MPM was shown to be more efficient, boasting a steeper convergence and higher accuracy.

The greater efficiency of MPM compared to commercial FEM software has similarly been proven in the context of traditionally challenging high-deformation problems; namely, Ma et al. [76] compared LS-DYNA's explicit FEM with the Material Point Method in the context of explosion problems, while Leroch et al. [73] applied this comparison to micro-milling simulations.

Fewer studies exist that compare the implicit MPM with other methods. A paper by Guilkey et al. [58] described a newly introduced implicit scheme for the MPM, and compared it with a commercial FEM code. While the results were consistent across the two methods, the Finite Element approach proved to be faster when applied to a billet compression problem; MPM is, however, expected to outperform its mesh-based counterpart in problems suffering from even larger deformations, as in this case expensive compromises such as remeshing would have to be adopted. Iaconeta et al. [56] compared the performance of an implicit MPM scheme with the Galerkin Meshfree Method (GMM) [77], proving how the implicit MPM shows superior robustness when dealing with large deformation problems.

3.4.4 Coupling with Other Methods

As mentioned previously, the MPM struggles to be competitive versus other mesh-based methods such as FEM in low deformation regimes. It is therefore natural to pair the two methods, using the former only in high deformation regions of the continuum. Zhang et al. [78] describes a Material Point Finite Element Method (MPFEM) which discretizes the body using a finite element mesh, adding a computational grid on top of predefined areas, which are predicted to suffer from high mesh distortion. This grid acts as the background mesh for the application of the Material Point Method.

Coupling strategies to pair MPM with SPH also exist [79]; once again, certain regions of the domain are prescribed to be treated with MPM and, in these areas, SPH particles are treated as material points. Employing SPH may be beneficial due to it being a legacy numerical scheme in CFD, with ample resources available; this coupling makes up for its inability to properly describe boundary conditions.

In the context of fluid-structure interactions, MPM has been coupled with a variety of different numerical methods, including the finite volume method [80, 81], the finite difference method [82], the finite element method [83, 84], and the hybrid immersed boundary method (HIBM) [85].

3.5 Applications

3.5.1 General applications

Large deformations. MPM was created to handle large deformation problems efficiently, where common mesh-based solutions, such as FEM, tend to fail. In particular, it was originally intended for highly dynamic problems such as contacts, impacts, penetration and perforation [28]. These problems are easily handled by MPM due to its ability to handle such problems natively, without requiring additional contact detection algorithms. Successful applications include welding and cutting processes [86, 87] and explosive blast and fragmentation problems [80, 88].

Geo-technical engineering. MPM's ability to deal with large strain problems, along with its flexibility when dealing with complex constitutive laws, have solidified the MPM as a premium choice in the geo-technical engineering field; a non-comprehensive list of applications include landslide simulations [89, 90, 91, 92], silo discharging [52, 53], anchor pull-out [93], excavator bucket filling [94] and pile driving [95, 96]. This method has proven to be competitive when compared to other popular choices in geomechanics, such as the discrete element method (DEM) [94]. A broader description of MPM applications in this field is available in the book by Fern et al. [97].

Fracture simulation. Modeling the initiation and propagation of fractures in a solid is a challenging problem in numerical simulations. Three approaches exist in the literature: discontinuous, continuous and mixed continuous-discontinuous; the MPM lends itself well for all three.

The discontinuous approach aims to model the crack formation explicitly, by separating the involved surfaces. In MPM, this amounts to allowing multiple velocity fields at nodes whose supports are cut by the cracks [98]. This method is overly expensive from a computational point of view.

On the other hand, a continuous approach only models the material degradation following a fracture. Multiple techniques are available to do so in MPM: one may either deactivate the fractured particles, by setting their deviatoric stresses to zero while still allowing them to contribute to the mass balance [99], or adopt an appropriately modified constitutive law that simulates damage [100].

Lastly, the mixed approach works by first introducing the aforementioned damage model, and then by applying a self-contact algorithm on the afflicted nodes to separate the material [101].

Image-based simulation. MPM (and, more generally, mesh-free methods) are well-suited for image-based simulations involving large displacements, as images can quite easily be converted into a set of particles, as opposed to a finite element grid; in particular, each pixel of the image can be converted into a material point [102, 103, 104, 105].

Computer graphics. Besides engineering applications, MPM has also been successfully employed in computer graphics; its rise to popularity is owed to its adoption by Walt Disney Animation Studios. Most notably, the foundational work that laid the path for such applications is that of Stomakhin et al. [72], which introduced a semi-implicit MPM scheme to simulate the behaviour of snow.

3.5.2 Applications in CFD

Despite it being first developed in a solid mechanics setting, MPM has always shown promising results when applied to both compressible and incompressible fluid dynamics. This section will delve into this topic in detail, and will be structured as follows: first, an introductory overview of its use cases in computational fluid dynamics will be provided, followed by a list of complications that uniquely characterize these applications; then, various MPM formulations within the context of fluid dynamics will be showcased, with greater attention being given to the mixed velocity-pressure MPM formulation that has been developed in this thesis.

Applications

Among the first results concerning compressible gas dynamics have been published by York et al. [4], which employed the Material Point Method to solve

a fluid-membrane interaction problem; such problems would otherwise require complex coupling between Eulerian and Lagrangian schemes. The unique advantages of MPM are especially evident here, such as the ability to use material points for both solid and fluid phases, and the built-in contact detection. A comprehensive review on fluid-structure interactions (FSIs) has been compiled by Hu et al. [106]. FSIs have also been modeled by coupling MPM with other numerical solvers, as described in Section 3.4.4. Further developments of gas dynamics in MPM are provided by Tran et al. [107]. Relevant advances in the topic of incompressible fluid dynamics have first been brought up by Li et al. [35], where a weakly compressible Material Point Method was defined in the setting of fluid sloshing. Several papers have focused on purely incompressible fluid dynamics [5, 9], with applications ranging from fluid-solid interactions [71, 108] to water flow through saturated porous media [34, 109, 110, 111, 112].

Complications

The main issues any MPM scheme has to address when applied to the dynamics of incompressible fluid flows are volumetric locking, quadrature errors and mapping errors; the latter have already been discussed in Section 3.4.2.

Volumetric locking is a well-documented issue that occurs when the incompressibility constraint is imposed [5, 39, 108, 113]. In particular, it is known to be especially prominent when lower-order basis functions are used. A wide array of solutions exist to circumvent or amend this issue. Employing higher order basis functions greatly mitigates locking issues; should one choose to use linear basis functions, several anti-locking algorithms have been developed, mainly based on the Hu-Washizu variational principle [108, 113]. Alternative approaches to implement truly incompressible equations of state include the fractional step method, first introduced to MPM by Kularathna and Soga [39] and Zhang et al. [5], and the mixed (or monolithic) approach, first used in MPM by Iaconeta et al. [7] and later extended to fluid dynamics by Chandra et al. [9].

Quadrature errors are, inevitably, one of the critical aspects of MPM, due to how the material points are used as integration points. Notably, however, material points tend to cluster along stream line in fluid flow simulations, exacerbating this phenomenon even further [114, 115]. The arising issues range from lower accuracy profiles to violations of the mass and momentum conservation; clustering of material points also leads to instability, as clustered particles carry an

excessively large volume, resulting in an overestimated pressure gradient force, which may cause the solution to explode. Particle clustering has been addressed using particle-shifting techniques inherited from SPH [70, 116].

3.6 MPM Formulations for Incompressible Fluid Dynamics

3.6.1 Review on Existing Methods

In this section, various approaches to handle fluid incompressibility in MPM will be showcased. The earliest attempts at implementing such problems opted for a weakly compressible equation of state, while true incompressibility has been achieved through a variety of different strategies; among them, both irreducible and mixed approaches will be discussed and compared.

Weakly Compressible MPM

Weak compressibility can be implemented by introducing an artificial equation of state which relates pressure to density [35]. Slight incompressibility is modeled as

$$\frac{\partial p}{\partial t} + \rho c^2 \nabla \cdot \boldsymbol{v} = 0, \tag{3.15}$$

where the artificial equation of state $p = \rho c^2$ has been introduced; c is a numerical sound speed, smaller than the physical speed of sound but at least an order of magnitude higher than the maximum fluid velocity.

While this approach is very intuitive to implement, it also carries some drawbacks: introducing a sound speed imposes an upper limit to the time step size, proportional to its magnitude; moreover, density fluctuations will be observed, which are, in turn, related to unphysical oscillations of the pressure, which would have to be numerically filtered out. Higher sound speeds limit these oscillations, so a compromise between accuracy and efficiency has to be struck.

Fractional-step Method

The fractional-step approach was originally introduced by Chorin [117] to solve irreducible formulations, and later ported to MPM by Zhang et al. [5] and Kularathna and Soga [39]. The fractional-step method works by splitting the

solution of the momentum equation in two steps: first, an intermediate velocity field is obtained by temporarily ignoring the pressure gradient contribution; then, the pressure terms are introduced to correct it.

The first step is performed by ignoring the pressure component in (2.12); the intermediate velocity $\boldsymbol{v}^*$ is then obtained by substituting the resulting σ into (2.8):

$$\boldsymbol{v}^* = \boldsymbol{v}^n + \frac{\delta t}{\rho}\left(\nabla \cdot \boldsymbol{\sigma}^{dev} + \boldsymbol{f}\right) \tag{3.16}$$

This is a linear problem which can be solved using explicit techniques.

Then, the intermediate velocity can be corrected using the pressure gradient:

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^* - \frac{\delta t}{\rho}\nabla p^{n+1} \tag{3.17}$$

Since the divergence-free condition (2.16) applies, the above equation yields a Poisson equation:

$$\nabla \cdot \boldsymbol{v}^* = \frac{\delta t}{\rho}\Delta p^{n+1} \tag{3.18}$$

It should, however, be noted that the stability of this method can be compromised by the imposition of pressure boundary conditions on the free surface; such instability leads to spurious oscillations in the solution fields. Pressure boundary conditions on the free surface can be imposed by applying a ghost fluid method [5] after defining the free surface with a level set approach; damping techniques are needed to address the arising spurious velocity modes.

Mixed Formulation

The mixed formulation is the most intuitive way to model incompressible fluid dynamics [1]. This approach is well-known in the FEM setting, where velocity and pressure fields are solved simultaneously as primary variables. Both the momentum and mass balance equations are thus solved together in a system of equations.

A mixed velocity-pressure formulation has been described in chapter 2. As mentioned there, the main drawback of such approach is the lack of stability, caused by the violation of the discrete inf-sup condition. A wide array of stabilization techniques have been developed to amend this issue in the FEM context, which

can be ported to MPM as well.

The use of mixed formulations in MPM is a relatively recent development, both in the field of solid mechanics [7, 8] and CFD [9]; furthermore, it has been employed in two-phase problems related to fluid flow in porous materials [111, 112].

A VMS-stabilized mixed formulation is adopted in Chandra et al. [9] and Moreno et al. [8]; in these works, however, a displacement-based approach is used instead, as is more common in MPM. This approach notably allows the fluid solver to easily be coupled with other materials in multi-phase simulations. On the other hand, solving the Navier-Stokes equations for displacement rather than velocity introduces additional complexity: while the operators of the weak form (2.49) are all linear, the same is not true for its displacement-based counterpart.

Mixed formulations are solved using monolithic approaches, which don't suffer from the imposition of pressure boundary conditions, unlike other solution strategies, like the popular fractional-step method.

3.6.2 Mixed VP MPM

The Material Point Method can be seen as a Finite Element scheme with an added initialization phase and followed by a convective phase, as described in Section 3.2. The FEM formulation described in the first chapter can thus be converted to MPM by enhancing it with these two phases.

Initialization phase. Mass, velocity and pressure at the previous timestep n are initially stored in the material points p, and are projected onto the background grid nodes i using the weighting functions ϕ_i:

$$m_i^n = \sum_{p=1}^{N_p} \phi_i(\boldsymbol{x}_p) m_p \tag{3.19}$$

$$\boldsymbol{v}_i^n = \frac{1}{m_i^n} \sum_{p=1}^{N_p} \phi_i(\boldsymbol{x}_p) m_p \boldsymbol{v}_p \tag{3.20}$$

$$\boldsymbol{a}_i^n = \frac{1}{m_i^n} \sum_{p=1}^{N_p} \phi_i(\boldsymbol{x}_p) m_p \boldsymbol{a}_p \tag{3.21}$$

UL-FEM phase. This phase is mostly as described in chapter 2. Once crucial difference, however, lies in how the integrals defining the various terms of the algebraic system are computed: while FEM uses Gaussian quadrature to ensure optimal precision, in MPM the material points themselves act as integration points, with their volumes acting as integration weights.

Another key difference between FEM and MPM at this stage concerns the spatial discretization: due to the hybrid nature of the latter, the computational domain is partitioned into elements, rather than the body itself (which is instead described as a set of particles), However, since, in the previous phase, the particle information have been mapped onto the computational grid, the two algebraic systems is solved as in FEM.

Convective phase. Once the solution has been obtained on the computational grid, it needs to be projected back to the particles:

$$\boldsymbol{u}_p^{n+1} = \sum_{i=1}^{N_n} \phi_i(\boldsymbol{x}_p)\boldsymbol{u}_i^{n+1} \tag{3.22}$$

$$\boldsymbol{a}_p^{n+1} = \sum_{i=1}^{N_n} \phi_i(\boldsymbol{x}_p)\boldsymbol{a}_i^{n+1} \tag{3.23}$$

$$p_p^{n+1} = \sum_{i=1}^{N_n} \phi_i(\boldsymbol{x}_p)p_i^{n+1} \tag{3.24}$$

As mentioned in Section 3.4.2, additional care is required when projecting velocities from the grid to the particles. One may choose to opt for a PIC scheme, using the grid velocities to interpolate the nodal values:

$$\boldsymbol{v}_p^{n+1} = \sum_{i=1}^{N_n} \phi_i(\boldsymbol{x}_p)\boldsymbol{v}_i^{n+1} \tag{3.25}$$

Alternatively, the FLIP approach computes the nodal velocities using the grid accelerations:

$$\boldsymbol{v}_p^{n+1} = \boldsymbol{v}_p^{n} + \boldsymbol{a}_p^{n+1}\delta t \tag{3.26}$$

Alternatively, a linear combination of the two schemes may also be adopted. In this work, the PIC scheme has been adopted. Lastly, after updating the material point positions using the displacement $\boldsymbol{u}_p^{n+1}$, the grid can be reset.

Algorithm

Algorithm 1 Algorithm for the VMS-stabilized mixed VP MPM

read initial condition $\boldsymbol{v}_p^0$
set $p_p^0 = 0$
compute $\boldsymbol{u}_p^0 = \boldsymbol{v}_p^0 \delta t$
for $n = 0, \ldots, t-1$ **do**
  clear nodal info and recover undeformed background mesh
  map previous time step's solution into nodes:
    $m_i = \sum_{p=1}^{N_p} \phi_i(\boldsymbol{x}_p) m_p$
    $\boldsymbol{u}_i^n = \frac{1}{m_i^n} \sum_{p=1}^{N_p} \phi_i(\boldsymbol{x}_p) m_p \boldsymbol{u}_p^n$
    $\boldsymbol{v}_i^n = \frac{1}{m_i^n} \sum_{p=1}^{N_p} \phi_i(\boldsymbol{x}_p) m_p \boldsymbol{v}_p^n$
    if $n > 0$ **then** $\boldsymbol{v}_i^{n-1} = \frac{1}{m_i^n} \sum_{p=1}^{N_p} \phi_i(\boldsymbol{x}_p) m_p \boldsymbol{v}_p^{n-1}$
  set FE functions $\boldsymbol{u}_h^n$, $\boldsymbol{v}_h^n$, p_h^n from nodal values $\boldsymbol{u}_i^n$, $\boldsymbol{v}_i^n$, p_i^n
  predictor of BDF2 method:
    $\delta \boldsymbol{v}_h^{n+1} = 0$
    $\delta p_h^{n+1} = 0$
    if $n = 0$ **then** $\boldsymbol{a}_h^{n+1} = \frac{1}{\delta t} \left(\boldsymbol{v}_h^{n+1} - \boldsymbol{v}_h^n \right)$
    if $n > 0$ **then** $\boldsymbol{a}_h^{n+1} = \frac{1}{2\delta t} \left(3\boldsymbol{v}_h^{n+1} - 4\boldsymbol{v}_h^n + \boldsymbol{v}_h^{n-1} \right)$
  set $k = 0$
  set $\delta \boldsymbol{v}_h^{n+1,k} = \delta \boldsymbol{v}_h^{n+1}$, $\delta p_h^{n+1,k} = \delta p_h^{n+1}$
  while not converged **do**
    $k \leftarrow k + 1$
    compute stabilization parameters: $\tau_1^{n+1,k}$, $\tau_2^{n+1,k}$
    evaluate local LHS and RHS
    assemble local terms to global LHS and RHS
    solve the algebraic system to obtain $\delta \boldsymbol{v}_h^{n+1,k}$ and $\delta p_h^{n+1,k}$
    corrector of BDF2 method:
      $\delta \boldsymbol{v}_h^{n+1,k} += \delta \boldsymbol{v}_h^{n+1,k-1}$
      $\delta p_h^{n+1,k} += \delta p_h^{n+1,k-1}$
    compute correction of $\boldsymbol{a}_h^{n+1}$:
      if $n = 0$ **then** $\boldsymbol{a}_h^{n+1} = \frac{1}{\delta t} \delta \boldsymbol{v}_h^{n+1,k}$
      if $n > 0$ **then** $\boldsymbol{a}_h^{n+1} = \frac{1}{2\delta t} \left(3\delta \boldsymbol{v}_h^{n+1,k} - \boldsymbol{v}_h^n + \boldsymbol{v}_h^{n-1} \right)$
    check convergence
  end while
  compute velocity, pressure and displacement:
    $\boldsymbol{v}_h^{n+1} = \delta \boldsymbol{v}_h^{n+1,k} + \boldsymbol{v}_h^n$
    $p_h^{n+1} = \delta p_h^{n+1,k} + p_h^n$
    $\boldsymbol{u}_h^{n+1} = \boldsymbol{u}_h^n + \delta t \boldsymbol{v}_h^{n+1}$
  set nodal values of velocity, pressure, displacement and acceleration:
    $\square_h^{n+1} = \square_h^{n+1,k}$
  interpolate nodal information into the material points:
    $\square_p^{n+1} = \sum_{i=1}^{N_n} \phi_i(\boldsymbol{x}_p) \square_i^{n+1}$
  update kinematics and material point positions (using $\boldsymbol{u}_p^{n+1}$)
end for

Chapter 4

Implementation in Kratos Multiphysics

The numerical scheme described in the previous chapters has been implemented, tested and verified in the framework *KRATOS Multiphysics*. A brief description of the software's structure will be provided in this chapter, as it is necessary to understand the details of this implementation.

4.1 Introduction to Kratos Multiphysics

KRATOS Multiphysics (Kratos) is an open-source framework[1] built for implementing parallel, high performance, multi-disciplinary simulation software. It was first created by the International Center of Numerical Methods in Engineering (CIMNE) [118, 119], and its original scope involved the implementation of finite element schemes, to be applied to a wide variety of engineering problems (eg. solid mechanics, fluid dynamics and thermodynamics) and to allow interactions between them. Support for other numerical methods, including the Material Point Method, has been added over time, thanks to its remarkable modularity and ease of extensibility.

[1] `https://github.com/KratosMultiphysics`

Figure 4.1: *Kratos Multiphysics* logo

4.1.1 Structure

Kratos is written in C++, but it's supplied with an extensive Python interface which is used to define the main procedure more flexibly. Its modularity is guaranteed by its object-oriented design, its multi-layer approach and its kernel and application structure.

The object-oriented design splits the problem into multiple objects, each with its own associated class; in Kratos, such objects have been created based on the FEM structure. A list of the main classes of Kratos is provided in figure 4.2.

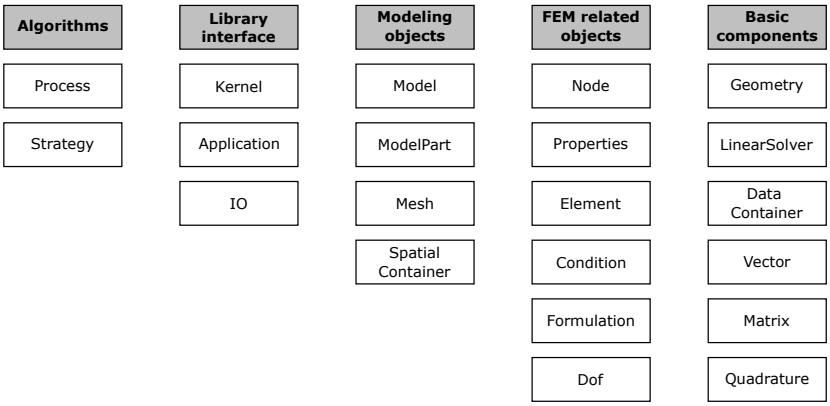


Figure 4.2: Main classes defined in Kratos (source: Dadvand et al. [118])

Among them, `Vector`, `Matrix` and `Quadrature` are defined by basic numerical concepts, `Node`, `Element`, `Condition` and `Dof` are inherited from FEM concepts, and `Model`, `ModelPart`, `Mesh` and `SpatialContainer` store geometrical data according to practical FEM methodology. The program flow is governed by `IO`, `LinearSolver`, `Process` and `Strategy`, while `Kernel` and `Application` are used for library management.

The multi-layer approach dictates how these objects interact, such that objects in a given layer can only interact with themselves or with objects in layers beneath them. This structure is crucial for reducing dependencies within the code, as well as increasing clarity. Objects are layered in increasing order of complexity, with basic input/output scripts on the outermost layer and core tools on the innermost one. Figure 4.3 shows how the layers are structured in Kratos.

The kernel and application approach allows different fields to be developed independently without conflicts; applications include their own custom classes, which are inherited from the core ones described above.

Many Kratos applications are also supplied with an optional user interface, available on GitHub[2], which attaches to the pre- and post-processing tool GiD[3], also developed by CIMNE.

[2] `https://github.com/KratosMultiphysics/GiDInterface`
[3] https://www.gidsimulation.com/

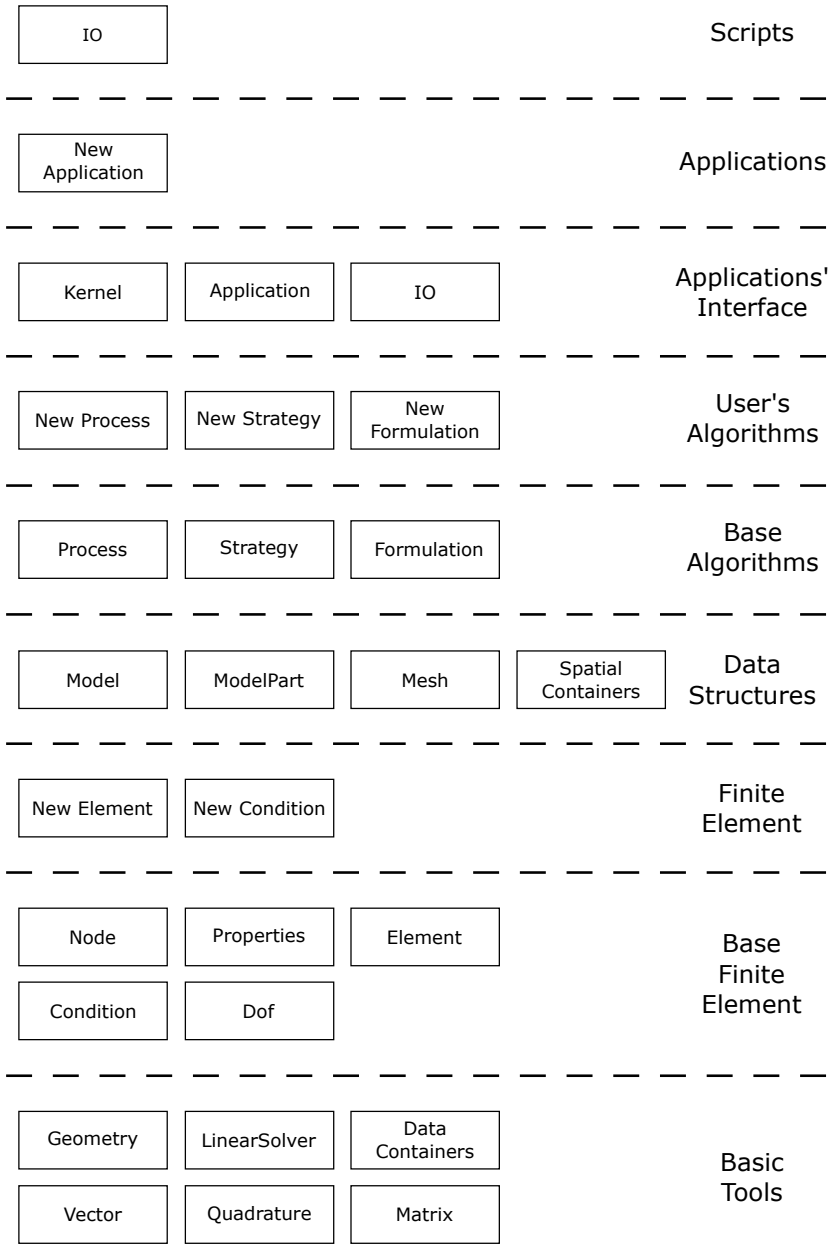


Figure 4.3: Layered structure of Kratos (source: Dadvand et al. [118])

4.2 Kratos MPM Application

This application, first developed by Iaconeta et al. [56] under the name "Particle Mechanics Application", implements the Material Point Method in Kratos[4]. This application introduces several custom classes; since the core classes have been developed in a FEM setting, they need to be adapted to the MPM environment. Some custom classes of particular importance for the purposes of the thesis are listed below.

Model Part. The core `ModelPart` class holds all data related to an arbitrary part of the model. This includes `Nodes`, `Properties`, `Elements` and `Conditions`, as well as the solution data. In `MPMApplication`, both the background grid geometry and the material point distribution have to be defined, and as such the `ModelPart` is further divided in three `SubModelParts`: `Background_Grid` stores the background grid and its boundary conditions, `Initial_MPM_Material` stores the mesh that generates the initial particle distribution, and `MPM_Material` stores the material points and the conditions after they've been copied from their respective `SubModelParts`.

Element. In `MPMApplication`, the `Element` class is used to define both the background grid elements, and the material points, which are used as moving integration points and which store density, volume and mass, along with material properties and kinematic variables.

Strategy. Custom `Strategy` classes are used to govern the simulation; the core `BuilderAndSolver` framework is used for this purpose, but the strategy is also supplied with MPM-specific operations such as mapping information to and from the material points.

Output. In the MPM setting, two different outputs may be of interest: either the solution variables at the backgroud grid, or at the material points. The core output classes are built to provide the results at the background grid, and thus custom classes are available to produce the results stored in the material points as well. Two approaches exist to visualize the results: the core class `GiDOutputProcess` and the custom class `MPMGiDOutputProcess` produce `.post.res`

[4] `https://github.com/KratosMultiphysics/Kratos/tree/master/applications/MPMApplication`

output files to be visualized within GiD itself; alternatively, the classes `VtkOutputProcess` and `MPMVtkOutputProcess` similarly store the results in `.vtk` files to be read using third-party tools.

4.3 Running an MPM Simulation

4.3.1 Pre-processing

Running a simulation on Kratos first requires some pre-processing steps to generate the necessary files. First of all, a main Python script is required to run the simulation.

Morevoer, the geometrical data used to generate both the background grid and the material point is stored in two `ModelPart` files (`.mdpa`). A third-party software is needed to generate these geometries.

Lastly, two `.json` files, which will henceforth be referred to as `ParticleMaterials.json` and `ProjectParameters.json` (though any other file name may be used), are used to store relevant parameters. `ParticleMaterials.json` only includes information on the material and its constitutive law, which is stored in the material points; `ProjectParameters.json` stores all other simulation-relevant information, such as solver settings, boundary conditions and loads, post-processing settings and various other entries.

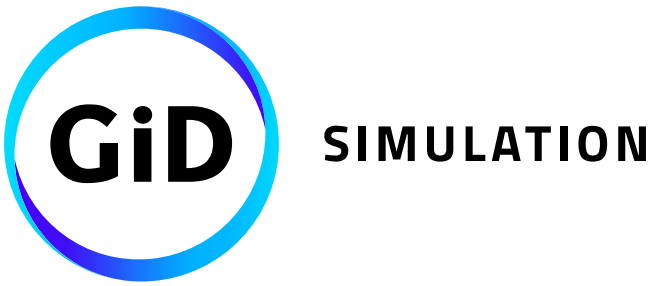


Figure 4.4: *GiD* logo

In this work, GiD's Kratos interface has been used for all pre-processing steps, since it's also compatible with `MPMApplication`. Body and grid meshes can readily be built in GiD, which also produces the appropriate `.json` files and a main Python script (`MainKratos.py`).

4.3.2 Workflow

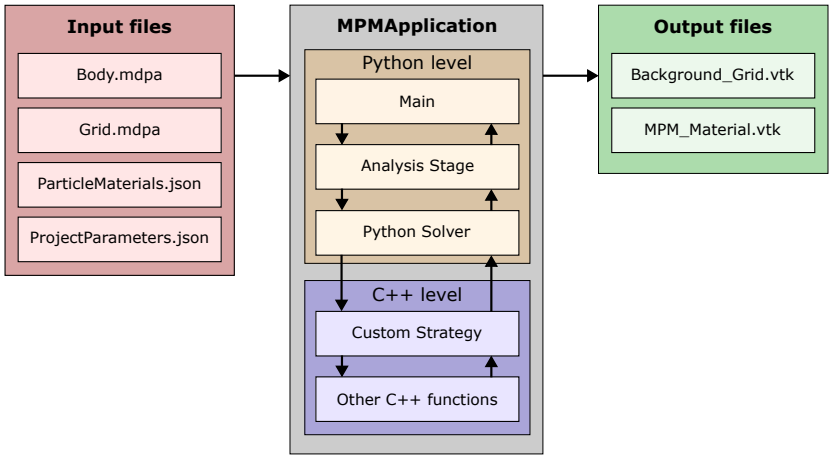


Figure 4.5: Workflow of `MPMApplication`

Figure 4.5 shows the workflow of `MPMApplication` as the simulation is ran. A brief description of each sequence will be provided here.

Main. The code is first ran from a main Python script, as mentioned in 4.3.1.

Analysis Stage. The major steps of the simulation (initialize, run, finalize) are called here.

Python solver. The information stored in `ProjectParameters` is read here and sent to the C++ level through the `pybind11` wrapper[5].

Strategy. On the C++ level, the simulation is managed here, by generating the material points according to the input data as well as calling all the relevant C++ codes.

Other C++ functions. The functions called by the scheme include custom elements and conditions.

[5]https://github.com/pybind/pybind11

4.3.3 Post-processing

For the purposes of this thesis, the open-source post-processing software ParaView[6] has been used to read the `.vtk` files and visualize the results of the simulations. To this end, Kratos produces two types of files for each timestep: an `MPM_Material_*.vtk` file, which stores the particle movements along with their respective material point variables, and a `Background_Grid_*.vtk` file, which stores the solution that was computed on the background grid during the Lagrangian phase. The former is often useful to understand the behaviour of the simulation over time, while the latter can be used to validate the results, comparing it to the results produced by other numerical schemes, such as FEM.

Figure 4.6: *ParaView* logo

4.4 Implementation

Implementing the mixed VP formulation in Kratos required work on multiple fronts. Among the various miscellaneous changes required throughout the `MPMApplication` files, the most relevant ones will be reported below.

4.4.1 Input Files

`MPMApplication` was built to only handle displacement-based formulations; thus, a new flag had to be added to the `ProjectParamters.json` file, in order to toggle on a velocity-based formulation instead; this flag, `velocity_formulation`, accepts boolean values and should be set to `true` to use the newly developed formulation.

Moreover, it was deemed convenient to have the arbitrary stabilization parameter c, detailed in (2.142), as an input parameter as well, despite current stabilized MPM schemes implemented in Kratos opting to have it as an internal parameter, not visible to the user. Moving this parameter to the input file allows it to be changed without the need of recompiling the code. Two constants have been added, through the entries `stabilization_constant_1` and `stabilization_-`

[6] https://www.paraview.org/

`constant_2`: despite the current implementation only needing one, this choice was made to allow greater flexibility, since many VMS implementations require two parameters instead.

No new entries have been added to the `ParticleMaterials.json` file; on the contrary, this numerical scheme makes no use of constitutive laws which are instead currently used in all other MPM implementations in Kratos. The absence of a `constitutive_law` parameter should therefore be noted.

4.4.2 Python Scripts

The `mpm_static_solver.py` and `mpm_solver.py` scripts of the Python interface have been modified to allow them to read the previously mentioned `velocity_formulation` flag in order to choose the correct time integration scheme, solution variable and element. Some C++ functions require knowledge of the value of the `velocity_formulation` flag; thus, `mpm_solver.py` also passes along its value through a newly added parameter, `IsMixedVP`.

4.4.3 Custom Elements

The element type is chosen in `material_point_generator_utility.cpp`; this file has thus been tweaked to read the previously mentioned `IsMixedVP` flag and choose the appropriate element type.

A new velocity-pressure (VP), VMS-stabilized mixed Lagrangian element has been developed: `MPMUpdatedLagrangianVPVMS`. Other custom elements currently implemented in MPM include: a Lagrangian Element, which inherits from the core class `Element`; a displacement-pressure (UP) mixed Lagrangian Element, which inherits from the Lagrangian one; and a VMS-stabilized UP Lagrangian Element, which inherits from its non-stabilized counterpart. Although rather similar in structure to the latter, the VP element cannot inherit functions from any of the aforementioned elements, as they are all displacement-based; thus, it inherits its base functions from `Element`.

The local components of the system matrix detailed in (2.151) are assembled here, while the global matrix is then obtained through standard core Kratos functions.

4.4.4 Custom Schemes

`MPMApplication` previously only offered an explicit time integration scheme, an implicit Newmark time integration scheme and a scheme for static cases; as for the preexisting custom elements, these schemes only support displacement-based formulations. A velocity-based static scheme (`MPMResidualBasedSimpleSteadyVelocityScheme`) has been written to validate the newly developed element under a simple, static regime, while the base algorithm required to develop a dynamic, velocity-based BDF2 time integration scheme has been written in 3.6.2, but it has yet to be translated to code.

Chapter 5

Validation Tests

The correctness and accuracy of the implemented numerical scheme has been checked by running several validation tests. The code has been tested on a static regime; the objective is to compare the numerical results it produces against a known analytical solution. By running each test multiple times, with increasingly finer meshes, a convergence plot can be drawn, as the numerical solution is expected to progressively converge towards the analytical one.

5.1 Convergence Rate

In order to evaluate convergence, a measure of the error committed by the numerical solution is needed. In this work, a weighted relative error has been used:

$$e_v(h) = \sqrt{\frac{\sum_{p=1}^{N_p} A_p ||\boldsymbol{v}_p^h - \boldsymbol{v}(\boldsymbol{x}_p)||_2^2}{\sum_{p=1}^{N_p} A_p ||\boldsymbol{v}(\boldsymbol{x}_p)||_2^2}} \tag{5.1}$$

$$e_p(h) = \sqrt{\frac{\sum_{p=1}^{N_p} A_p ||p_p^h - p(\boldsymbol{x}_p)||_2^2}{\sum_{p=1}^{N_p} A_p ||p(\boldsymbol{x}_p)||_2^2}} \tag{5.2}$$

where $||\boldsymbol{x}||_2^2 = \sum_{i=1}^{d} |x_i|^2$ is the Euclidean (L^2) norm.

The convergence rate is related to the error of the numerical solution through

the relation

$$e(h) \approx Ch^k, \tag{5.3}$$

with h being the mesh size, C being a positive constant and k being the convergence rate ($k = 1$ for linear convergence, $k = 2$ for quadratic convergence, etc.); plotting the data on a log-log plot, with mesh size on the x-axis and error norm on the y-axis, allows for easy evaluation of k as the slope of the plotted line.

The theoretical convergence rate for the velocity is quadratic ($k = 2$), as stated by the Nitsche-Aubin theorem.

Theorem 5.1.1 (Nitsche-Aubin)**.** Let $\boldsymbol{v}$ be the solution to the weak form (2.49) and $\boldsymbol{v}_h$ be the solution to the discretized weak form (2.61) with P_1 linear functions. Then:

$$e(h) = ||\boldsymbol{v} - \boldsymbol{v}_h|| \leq h^2 ||\boldsymbol{v}''|| \tag{5.4}$$

□

Remark 5.1.2. Due to the relation (2.102) between the function spaces $\mathcal{V}_h$ and $\mathcal{Q}_h$, 5.1.1 also implies linear convergence of the pressure ($k = 1$).

5.2 Method of Manufactured Solutions

The method of manufactured solutions (MMS) is a technique to validate a numerical code [120]. It works by choosing an *a priori* velocity and pressure field; the prescribed (or manufactured) solution fields are used to analytically compute the external forces that would generate it, as well as the boundary conditions (which must coincide with the manufactured solutions at the boundaries). When executing the code, the obtained numerical solution is expected to converge towards the manufactured solution for increasingly finer meshes. The method of manufactured solutions was first used in an MPM setting by Wallstedt and Guilkey [60], in order to evaluate the convergence of the newly created GIMP method. In the context of mixed MPM formulations, MMS has been employed to validate a VMS-stabilized displacement-pressure formulation by Moreno et al. [8].

5.2.1 Manufactured Solutions

The code has been tested with both a linear and a sinusoidal manufactured solution in a static regime.

The chosen linear manufactured velocities and pressure are:

$$\begin{cases} \boldsymbol{v}_x^{\text{lin}} = 4x + 6 \\ \boldsymbol{v}_y^{\text{lin}} = -4y - 6 \\ p^{\text{lin}} = x \end{cases} \tag{5.5}$$

According to (2.20), the external forces generating such solution field are:

$$\begin{cases} f_x^{\text{lin}} = 1 \\ f_y^{\text{lin}} = 0 \end{cases} \tag{5.6}$$

The code has also been tested with a more complex, nonlinear solution by imposing sinusoidal velocities and pressure:

$$\begin{cases} \boldsymbol{v}_x^{\text{sin}} = \sin(\pi x)\sin(\pi y) \\ \boldsymbol{v}_y^{\text{sin}} = \sin(\pi x)\sin(\pi y) \\ p^{\text{sin}} = \sin(\pi x) + \sin(\pi y) \end{cases} \tag{5.7}$$

The corresponding external forces then are:

$$\begin{cases} f_x^{\text{sin}} = -\pi\mu\left(\cos(\pi x)\sin(\pi y) + \sin(\pi x)\cos(\pi y)\right) + \pi\cos(\pi x) \\ f_y^{\text{sin}} = -\pi\mu\left(\cos(\pi x)\sin(\pi y) + \sin(\pi x)\cos(\pi y)\right) + \pi\cos(\pi y) \end{cases} \tag{5.8}$$

5.2.2 Simulation Settings

Material properties. The body simulated by these tests is a generic incompressible Newtonian fluid with density $\rho = 1.0\ \text{kg/m}^3$ and dynamic viscosity $\mu = 1.0\ \text{kg/(m}\cdot\text{s)}$.

Geometry. The computational domain has been taken as a 1.0×1.0 m square, discretized using structured triangular elements. 3 material points per node have been assigned, located at the Gauss points, with the fluid covering the whole computational domain. Boundary conditions are imposed on all boundaries of

the computational domain, prescribing velocity and pressure ad described earlier in this section. The tests have been ran with varying characteristic element sizes:

Element size [m]	
h_0	0.2
h_1	0.1
h_2	0.05
h_3	0.025

Table 5.1: Characteristic element sizes h.

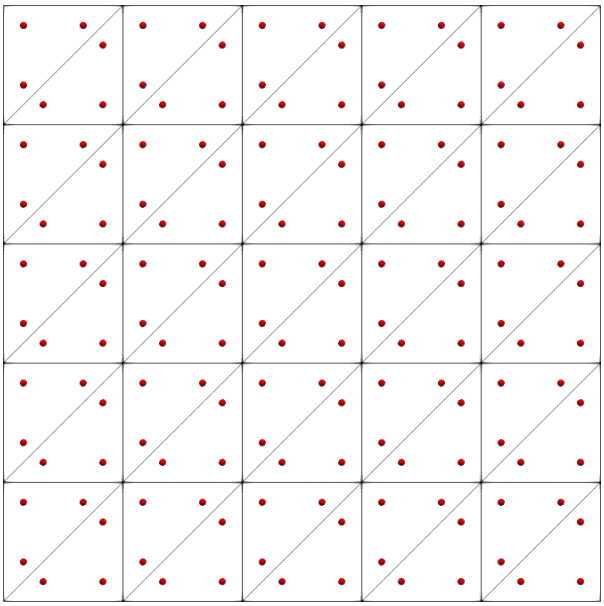


Figure 5.1: Example of material point distribution with element size $h = 0.2$ m

Launch script. Though GiD generates its own `MainKratos.py` file to run the simulation, a new main script, tailored to the needs of this test, has been written. The manufactured solution is imposed on the boundaries, and the associated external forces are imposed on the whole domain; after the simulation is ran for increasingly finer meshes, the relative convergence errors are computed and printed on a logarithmic plot.

Stabilization parameters. The arbitrary constant c dictating the magnitude of the stabilization parameter τ_1 has been set to $c = 0.01$; this value should be

high enough to guarantee convergence, but low enough to reduce the variational crime to a minimum.

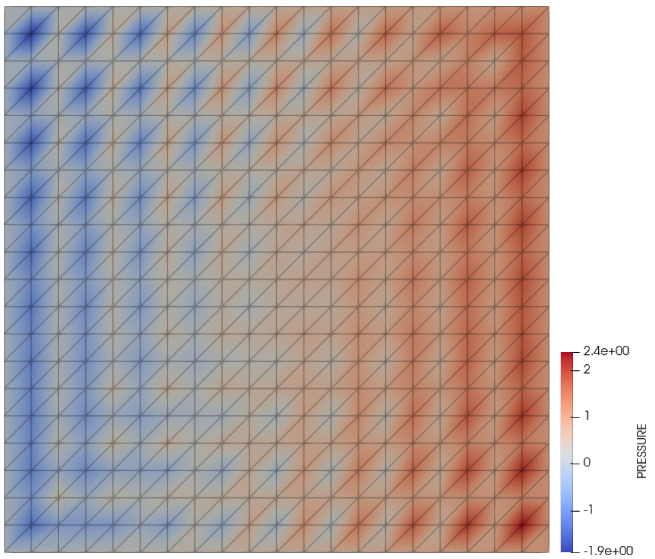


Figure 5.2: Unstabilized pressure field of the linear MMS solution; note that, as stated in 2.7, spurious modes in the solutions only show up in the pressure field.

5.2.3 Results

Linear solution.

h [m]	e_v	Velocity error ratio	e_p	Pressure error ratio
0.2	$1.34e-05$	—	$2.14e-03$	—
0.1	$2.63e-06$	5.08	$6.47e-04$	3.30
0.05	$5.56e-07$	4.72	$1.78e-04$	3.65
0.025	$1.27e-07$	4.39	$4.66e-05$	3.81

Table 5.2: Convergence rates of velocity and pressure for the linear MMS solution

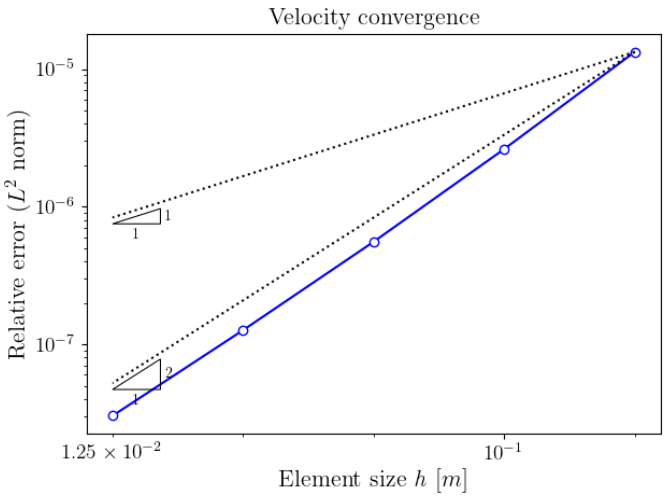


Figure 5.3: Logarithmic velocity convergence plot of the linear MMS test, evaluated using the L^2 norm relative error.

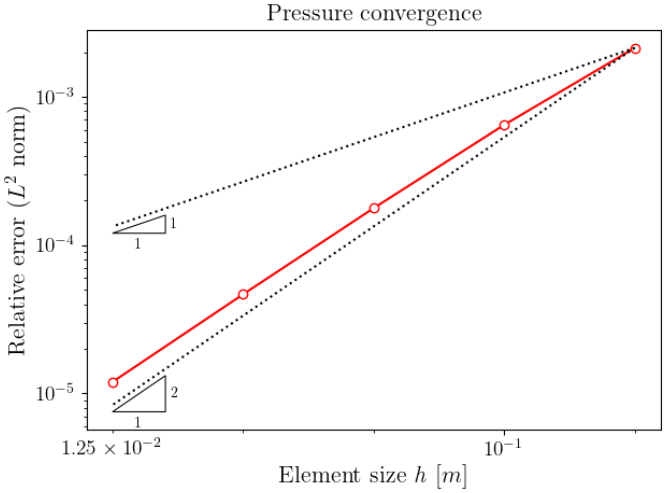


Figure 5.4: Logarithmic pressure convergence plot of the linear MMS test, evaluated using the L^2 norm relative error.

Sinusoidal solution.

h [m]	e_v	Velocity error ratio	e_p	Pressure error ratio
0.2	$3.19e-04$	—	$1.56e-03$	—
0.1	$8.37e-05$	3.81	$4.49e-04$	3.46
0.05	$2.07e-05$	4.04	$1.24e-04$	3.62
0.025	$5.10e-06$	4.06	$3.31e-05$	3.75

Table 5.3: Convergence rates of velocity and pressure for the sinusoidal MMS solution

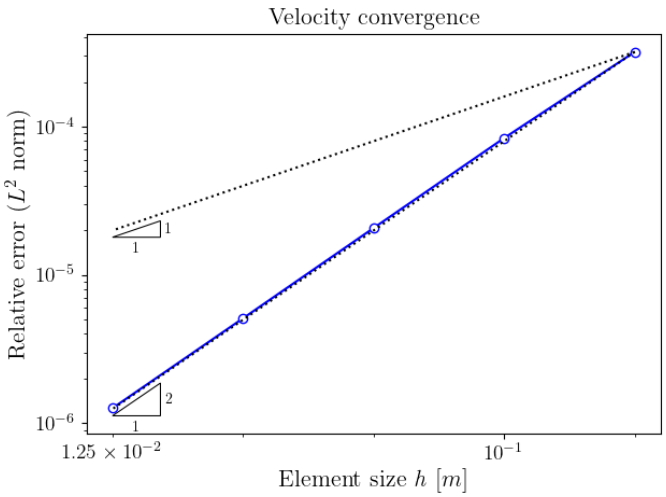


Figure 5.5: Logarithmic velocity convergence plot of the sinusoidal MMS test, evaluated using the L^2 norm relative error.

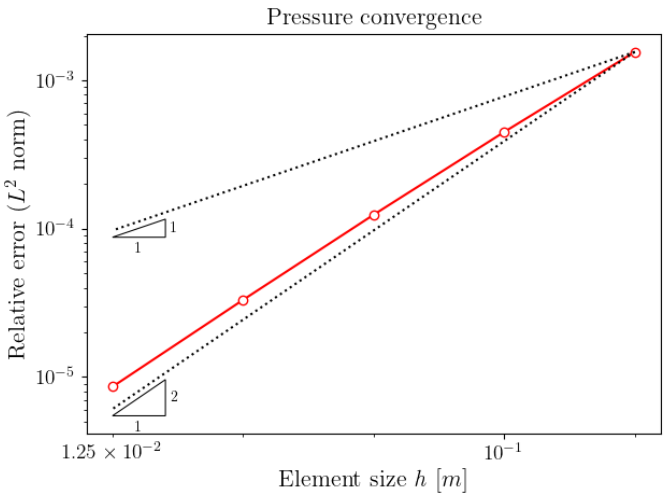
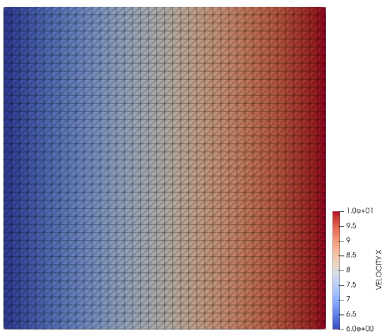
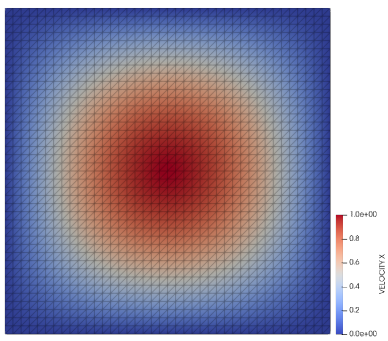


Figure 5.6: Logarithmic pressure convergence plot of the sinusoidal MMS test, evaluated using the L^2 norm relative error.

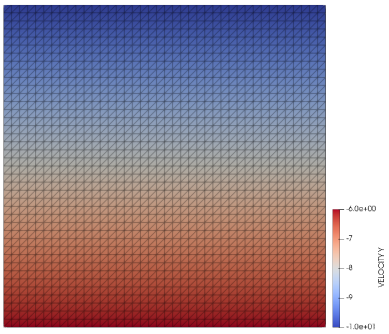
Tables 5.2 and 5.3 report the convergence rates of both variables for the linear and sinusoidal manufactured solution, respectively. Both velocity and pressure converge as the mesh is refined; moreover, it can be noticed that the rate of convergence of the pressure is of order two, thus being greater than the expected linear convergence. This phenomenon, known as superconvergence, can sometimes occur when exceedingly regular meshes and/or solutions are employed.



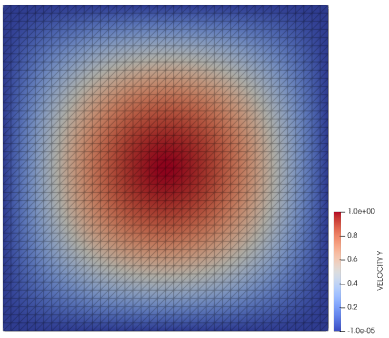
(a) Linear velocity field (v_x)



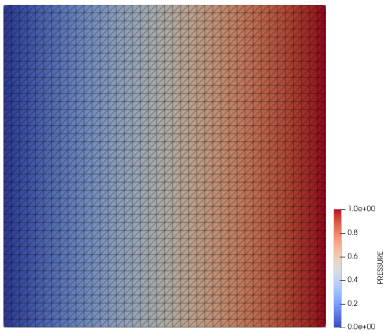
(d) Sinusoidal velocity field (v_x)



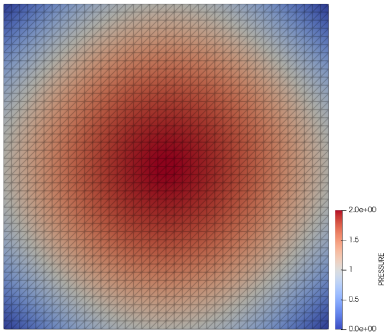
(b) Linear velocity field (v_y)



(e) Sinusoidal velocity field (v_y)



(c) Linear pressure field (p)



(f) Linear pressure field (p)

Figure 5.7: Vector fields of the numerical solution, with $h = 0.025$ m. The linear solution (5.5) is reported on the left, and the sinusoidal one (5.7) is on the right.

5.3 Still Water Test

5.3.1 Problem Description

The still water problem simulates a water well, described by a rectangular two-dimensional computational domain, subject to gravity. No-slip boundary conditions have been imposed on the three walls of the well, while on the water surface the pressure is fixed to the atmospheric pressure (atmospheric pressure is set to zero for convenience).

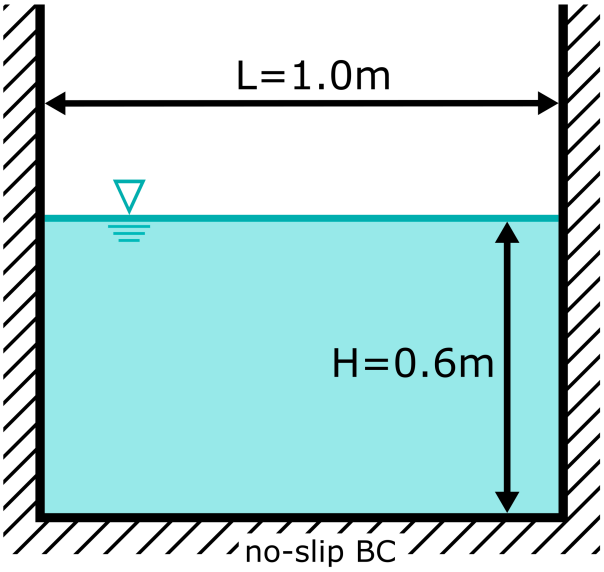


Figure 5.8: Schematic diagram of the still water test.

The expected behaviour of the pressure is described by the equation

$$p = \begin{cases} 0 & \text{if } y > H \\ \rho g(H - y) & \text{if } 0 \leq y \leq H \end{cases} \tag{5.9}$$

where H is the water level, ρ is the water density and g is the standard acceleration of gravity.

5.3.2 Simulation Settings

Material properties. The fluid simulated by this test has the properties of water, with density $\rho = 1000$ kg/m^3 and dynamic viscosity $\mu = 1.0{\cdot}10^{-3}$ kg/(m$\cdot$s)

Geometry. The computational domain coincides with the 1.0×1.0 m water well, and is discretized using structured triangular elements. 3 material points per node have been assigned, located at the Gauss points. The fluid covers a portion of the domain of size $1.0 \times H$ m, where the water level H has been set to $H = 0.6$ m in order to evaluate the behaviour of the water-air interface, while the pressure has been imposed to zero on the free surface. No-slip boundary

conditions have been imposed on the walls of the well. The simulations have been ran with the following element sizes:

Element size [m]	
h_0	0.2
h_1	0.1
h_2	0.05
h_3	0.025
h_4	0.0125

Table 5.4: Characteristic element sizes h.

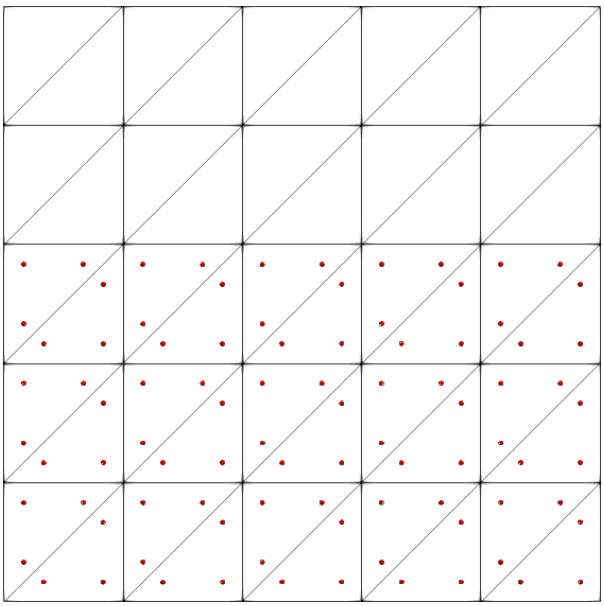


Figure 5.9: Material point distribution with element size $h = 0.2$ m and water level $H = 0.4$ m.

Launch script. The simulation is ran using the main script produced by GiD's pre-processing, while the logarithmic convergence curves have been drawn using the `matplotlib` Python library.

Stabilization parameters. The arbitrary constant c has been set to $c = 0.01$, as previously determined in the MMS test.

5.3.3 Results

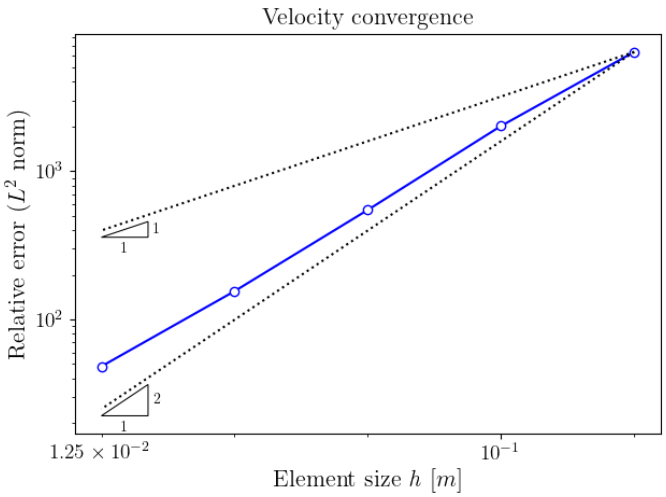


Figure 5.10: Logarithmic velocity convergence plot of the still water test, evaluated using the L^2 norm relative error.

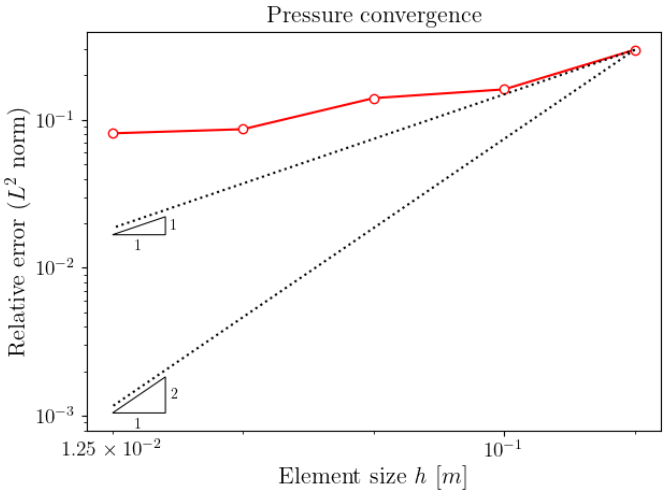


Figure 5.11: Logarithmic pressure convergence plot of the still water test, evaluated using the L^2 norm relative error.

h [m]	e_v	Velocity error ratio	e_p	Pressure error ratio
0.2	$6.47\text{e}-11$	—	$1.62\text{e}-08$	—
0.1	$1.79\text{e}-11$	3.61	$1.77\text{e}-08$	0.92
0.05	$5.85\text{e}-12$	3.06	$2.02\text{e}-08$	0.87
0.025	$2.65\text{e}-12$	2.02	$2.31\text{e}-08$	0.88
0.0125	$6.21\text{e}-13$	4.26	$2.47\text{e}-08$	0.93

Table 5.5: Convergence rates of velocity and pressure for the still water test

Looking at the results above, it would seem like the pressure is not converging to the analytical solution as fast as expected. A closer look at the solution, however, reveals why this is happening: it is likely that the linear nature of the solution, coupled with the structured grid and the linear basis functions, leads to a satisfactory result even on the roughest mesh, thus preventing an appropriate convergence curve to manifest. Thus, the lack of convergence reported in table 5.5 does not correlate to a lack of precision in the numerical scheme.

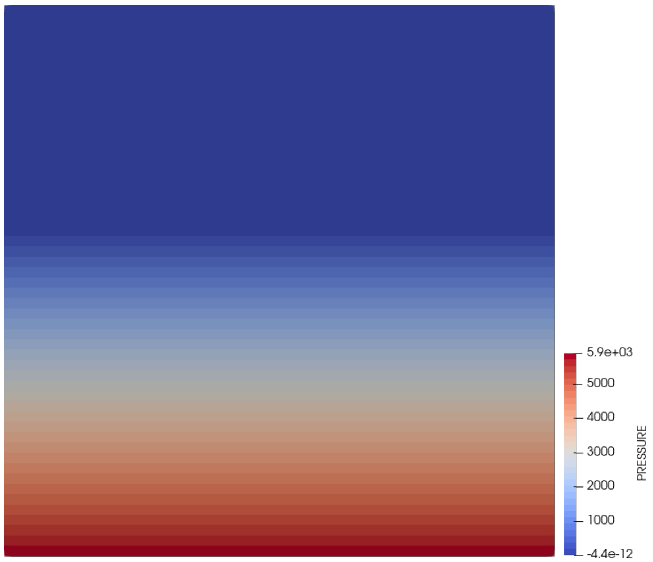


Figure 5.12: Pressure distribution of the still water problem.

5.4 Poiseuille Flow

5.4.1 Problem Description

Poiseuille flow is a stationary, pressure-induced fluid flow between two stationary and parallel plates. The flow is uniform and unidirectional, meaning that it is steady ($\partial_t v_x = 0$) and homogeneous along the longitudinal direction ($\partial_x v_x = 0$). The resulting velocity profile would be such that $\boldsymbol{v} = (v_x(y), 0)$; to obtain this result, the one-dimensional, homogeneous Navier-Stokes equation along the x-axis is used:

$$-\mu \frac{\partial v_x^2}{\partial^2 y} + \frac{\partial p}{\partial x} = 0 \tag{5.10}$$

Assuming the plates are fixed at heights $y = 0$ and $y = H$ respectively, that the no-slip condition is imposed on both plates ($\boldsymbol{v}(0) = 0$, $\boldsymbol{v}(H) = 0$), and that the flow is induced by a constant pressure gradient $G := -\partial_x p$, the resulting velocity profile is then expressed as:

$$v_x = \frac{G}{2\mu} y(H - y), \tag{5.11}$$

which will attain its maximum v_0 at $y = H/2$.

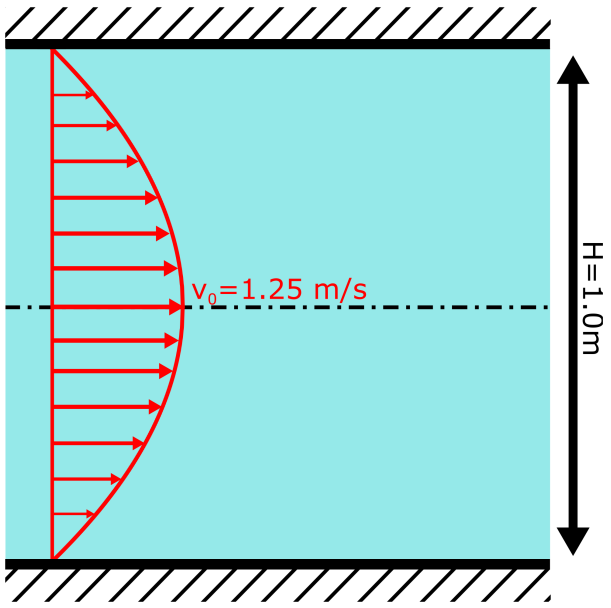


Figure 5.13: Schematic diagram of the Poiseuille flow.

5.4.2 Simulation Settings

Material properties. The fluid simulated by this test has the properties of water, with density $\rho = 1000$ kg/m^3 and dynamic viscosity $\mu = 1.0 \cdot 10^{-3}$ kg/(m$\cdot$ s)

Geometry. The computational domain has been taken as a 10.0 m long section of the region between the plates, in order to account for the assumption of infinitely long channel; the distance between plates is $H = 1.0$ m. A pressure gradient $G = 100$ Pa/m is imposed, which should lead to a parabolic velocity profile whose maximum $v_0 = 1.25$ m/s should be attained in $y = H/2$. The domain is discretized using structured triangular elements, with 3 material points assigned per node, located at the Gauss points. No-slip boundary conditions are imposed on the plates; the expected velocity profile is imposed on the inlet, while an arbitrary pressure value $p = 0$ Pa is imposed on the outlet. The simulations have been ran with the following element sizes:

Element size [m]	
h_0	0.2
h_1	0.1
h_2	0.05
h_3	0.025
h_4	0.0125

Table 5.6: Characteristic element sizes h.

Launch script. The same script used in for the still water test has been employed to run this problem and plot its results.

Stabilization parameters. The arbitrary constant c has been set to $c = 0.01$, as in previous tests.

5.4.3 Results

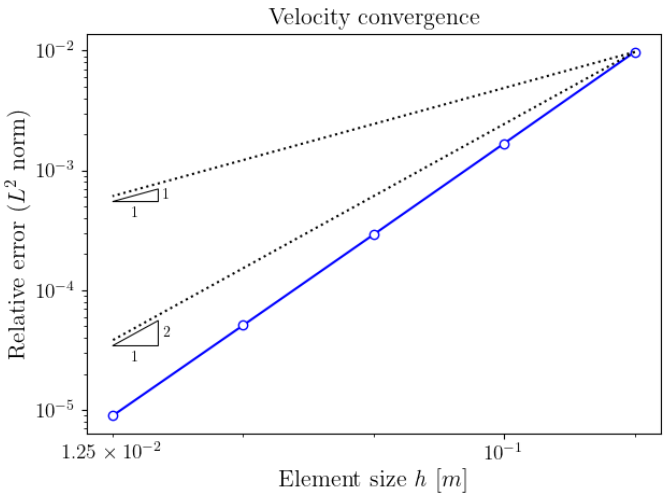


Figure 5.14: Logarithmic velocity convergence plot of the Poiseuille flow test, evaluated using the L^2 norm relative error.

h [m]	e_v	Velocity error ratio
0.2	9.76e − 03	—
0.1	1.68e − 03	5.81
0.05	2.92e − 04	5.77
0.025	5.09e − 05	5.73
0.0125	8.94e − 06	5.70

Table 5.7: Convergence rates of the velocity for the Poiseuille flow test

The data reported in table 5.7 shows that, similarly to the well balance test, the appropriate rate of convergence for the velocity is also attained for the Poiseuille flow test.

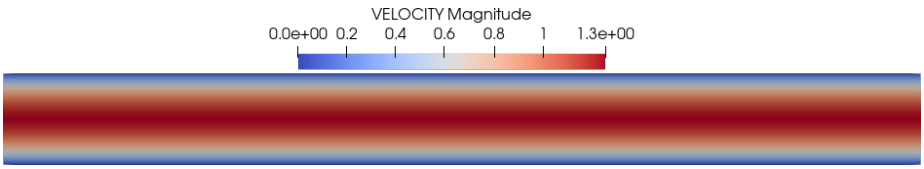


Figure 5.15: Velocity distribution of the Poiseuille flow.

Chapter 6

Conclusions and Outlook

6.1 Conclusions

In this work, a novel mixed Material Point Method (MPM) formulation for the Stokes problem has been formulated, adopting velocity and pressure as main dependent variables, as is more common in a Finite Element setting.

To this end, the balance equations governing the incompressible, Newtonian fluid flow have been studied in Chapter 2 and a Finite Element weak formulation of the problem has been obtained. Special care has been paid when stabilizing the algebraic system, as the Stokes problem is prone to instabilities in the pressure field; a Variational Multiscale (VMS) stabilization has been formulated by performing a Fourier analysis of the problem.

An in-depth review of the Material Point Method has been laid out in Chapter 3. The general structure and formulations of MPM has been described, along with its current use cases, strengths and weaknesses, in order to paint a general picture as to why one may want to adopt this solution over other popular numerical methods, with special focus on its applications in computational fluid dynamics (CFD); it was concluded that, while the topic of mixed formulations in MPM is not well fleshed out, it can prove itself to be a valid choice, given the MPM's superior performance when compared to legacy methods such as SPH. The steps required to develop the new MPM formulation have then been listed; an algorithm of the numerical scheme has been provided, to guide its

implementation via code.

The static version of the scheme has been successfully implemented in the open-source framework *Kratos Multiphysics*; while a number of different MPM formulations were already included in the software before this work, none of them supported the use of velocity as a main variable. In Chapter 4, the details of this implementation have been provided, after an introduction to the multi-layered, object-oriented structure of *Kratos*.

The implemented scheme has been validated and tested in Chapter 5. The Method of Manufactured Solutions (MMS) has been used to test it against generic solutions, of both linear and nonlinear nature; the code displayed satisfactory levels of convergence in both cases, for both dependent variables. The stabilizing effects of VMS have also been shown to be adequate to eliminate spurious modes in the pressure. The code has also been tested on a physics-based scenario, the still water test, where good levels of convergence have been achieved for the velocity; due to the linear nature of the pressure distribution, however, this test has proven inadequate to verify how optimally the pressure converges. The appropriate convergence of the velocity has then been further confirmed by running a Poiseuille flow simulation.

6.2 Outlook

While the scheme has only been implemented in its static version, the algorithm detailing a dynamic implementation has been provided in this work; future works may involve the addition of this scheme to *Kratos*. Moreover, further studies could be performed to evaluate the behaviour of non-homogeneous boundary conditions. The scheme's output may be compared with the results of a commercial Finite Element scheme to evaluate its performance, as well as with the displacement-pressure formulation currently implemented in *Kratos*. The results of this work may be fit to be published in an academic paper, to showcase the validity of this new formulation.

Bibliography

[1] J.N. Reddy and D.K. Gartling. *The Finite Element Method in Heat Transfer and Fluid Dynamics*. CRC Press, 2010.

[2] Franco Brezzi and Michel Fortin. *Mixed and Hybrid Finite Element Methods*. Springer New York, NY, 1991.

[3] Ivo Babuška. Error-bounds for finite element method. *Numerische Mathematik*, 16:322–333, 1971.

[4] Allen R. York II, Deborah Sulsky, and Howard L. Schreyer. Fluid-membrane interaction based on the material point method. *International Journal for Numerical Methods in Engineering*, 48:901–824, 2000.

[5] Fan Zhang, Xiong Zhang, Kam Yim Sze, Yanping Lian, and Yan Liu. Incompressible material point method for free surface flow. *Journal of Computational Physics*, 330:92–110, 2017.

[6] Shyamini Kularathna. *Splitting solution scheme for material point method*. PhD thesis, University of Cambridge, Department of Engineering, 2018.

[7] Ilaria Iaconeta, Antonia Larese, Riccardo Rossi, and Eugenio Oñate. A stabilized mixed implicit material point method for non-linear incompressible solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 368:1243–1260, 2019.

[8] Laura Moreno, Roland Wuechner, and Antonia Larese. A mixed stabilized mpm formulation for incompressible hyperelastic materials using variational subgrid-scales. *Computer Methods in Applied Mechanics and Engineering*, 435, 2025.

[9] Bodhinanda Chandra, Ryota Hashimoto, Shinnosuke Matsumi, Ken Kam-

rin, and Kenichi Soga. Stabilized mixed material point method for incompressible fluid flow analysis. *Computer Methods in Applied Mechanics and Engineering 419*, 2024.

[10] Leopoldo P. Franca, Thomas J.R. Hughes, Abimael F.D. Loula, and Miranda Isidoro. A new family of stable elements for nearly incompressible elasticity based on a mixed petrov-galerkin finite element formulation. *Numerische Mathematik*, 53:123–141, 1988.

[11] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2003.

[12] Sergio Rodolfo Idelsohn, Eider Oñate, and Facundo Del Pin. The particle finite element method; a powerful tool to solve incompressible flows with free-surfaces and breaking waves. *International Journal For Numerical Methods in Engineering*, 61:964–989, 2004.

[13] C.W. Hirt, A.A. Amsden, and J.L Cook. An arbitrary lagrangian-eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14:227–253, 1974.

[14] Charles S. Peskin. The immersed boundary method. *Acta Numerica*, 11:227–253, 1974.

[15] L.B. Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024, 1977.

[16] R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

[17] Damien Violeau and Benedict D. Rogers. Smoothed particle hydrodynamics (sph) for free-surface flows: past, present and future. *Journal of Hydraulic Research*, 54:1–26, 2016.

[18] Hitoshi Gotoh and Abbas Khayyer. On the state-of-the-art of particle methods for coastal and ocean engineering. *Coastal Engineering Journal*, 60:79—-103, 2018.

[19] Francis H. Harlow. The particle-in-cell method for numerical solution of problems in fluid dynamics. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 1962.

[20] J.U. Brackbill and H.M. Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65:314–343, 1986.

[21] J.U. Brackbill, D.B. Kothe, and H.M. Ruppel. Flip: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48:25–38, 1988.

[22] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics*, 34:1–10, 2015.

[23] Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. A polynomial particle-in-cell method. *ACM Transactions on Graphics*, 34:1–12, 2017.

[24] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Transactions on Graphics*, 24:965–972, 2005.

[25] Wen Zheng, Bo Zhu, Byungmoon Kim, and Ronald Fedkiw. A new incompressibility discretization for a hybrid particle mac grid representation with surface tension. *Journal of Computational Physics*, 280:96–142, 2015.

[26] Jian Zhu, You Quan Lui, Yuan Zhang Chang, and En Hua Wu. Animating turbulent water by vortex shedding in pic/flip. *Science China Information Sciences*, 56:1–11, 2013.

[27] Francis H. Harlow and Eddie J. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8:2182–2189, 1965.

[28] Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87:236–252, 1995.

[29] C. Taylor and P. Hood. A numerical solution of the navier-stokes equations using the finite element technique. *Computers Fluids*, 1:73–100, 1973.

[30] Ferdinando Auricchio, Franco Brezzi, and Carlo Lovadina. Chapter 9 mixed finite element methods. *Encyclopedia of Computational Mechanics*, 2004.

[31] D.N. Arnold, F. Brezzi, and M. Fortin. A stable finite element for the

stokes equations. *CALCOLO*, 21:337–344, 1984.

[32] Thomas J.R. Hughes, Leopoldo P. Franca, and Gregory M. Hulbert. A new finite element formulation for computational fluid dynamics: Viii. the galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73:173–189, 1989.

[33] Thomas J.R. Hughes, Gonzalo R. Feijóo, Luca Mazzei, and Jean-Baptiste Quincy. The variational multiscale method—a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166:3–24, 1998.

[34] Yuya Yamaguchi, Shinsuke Takase, Shuji Moriguchi, and Kenjiro Terada. Solid–liquid coupled material point method for simulation of ground collapse with fluidization. *Computational Particle Mechanics*, 7:209–223, 2020.

[35] J.G. Li, Y. Hamamoto, Y. Liu, and X. Zhang. Sloshing impact simulation with material point method and its experimental validations. *Computers Fluids*, 103:86–99, 2014.

[36] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, 2005.

[37] Ted Belytschko, Wing Kam Liu, Brian Moran, and Khalil I. Elkhodary. *Nonlinear Finite Elements for Continua and Structures*. John Wiley Sons, 2000.

[38] Thomas J.R. Hughes, Leopoldo P. Franca, and Marc Balestra. A new finite element formulation for computational fluid dynamics: V. circumventing the babuška-brezzi condition: a stable petrov-galerkin formulation of the stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering*, 59:85–99, 1985.

[39] Shyamini Kularathna and Kenichi Soga. Implicit formulation of material point method for analysis of incompressible materials. *Computer Methods in Applied Mechanics and Engineering*, 313:673–686, 2017.

[40] Rutherford Aris. *Vectors, Tensors, and the Basic Equations of Fluid Mechanics*. Dover Publications, 1962.

[41] Franco Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. *R.A.I.R.O. Analyse*

Numérique, 8:129–151, 1974.

[42] Thomas J.R. Hughes. Multiscale phenomena: Green's functions, the dirichlet-to-neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering*, 127:387–401, 1995.

[43] F. Brezzi, L.P. Franca, T.J.R. Hughes, and A. Russo. $b = \int g$. *Computer Methods in Applied Mechanics and Engineering*, 145:329–339, 1996.

[44] Ramon Codina, Javier Principe, Oriol Guasch, and Santiago Badia. Time dependent subscales in the stabilized finite element approximation of incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 196:2413–2430, 2007.

[45] Ramon Codina. A stabilized finite element method for generalized stationary incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 190:2681–2706, 2001.

[46] Ramon Codina. Stabilization of incompressibility and convection through orthogonal sub-scales in finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 190:1579–1599, 2000.

[47] Ramon Codina. Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. *Computer Methods in Applied Mechanics and Engineering*, 191:4295–4321, 2002.

[48] Deborah Sulsky, Zhen Chen, and Howard L. Schreyer. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, 118:179–196, 1994.

[49] S.G. Bardenhagen and E.M. Kober. The generalized interpolation material point method. *Computer Modeling in Engineering Sciences*, 5:477–495, 2004.

[50] A. Sadeghirad, R.M. Brannon, and J. Burghardt. A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations. *International Journal for Numerical Methods in Engineering*, 86:1435–1456, 2011.

[51] Alban de Vaucorbeil, Vinh Phu Nguyen, and Christopher R: Hutchinson. A total-lagrangian material point method for solid mechanics problems involving large deformations. *Computer Methods in Applied Mechanics*

and Engineering, 360, 2019.

[52] Zdzisław Więckowski, Sung-Kie Youn, and Jeoung-Heum Yeon. A particle-in-cell solution to the silo discharging problem. *International Journal for Numerical Methods in Engineering*, 45:1203–1225, 1999.

[53] Zdzisław Więckowski. The material point method in large strain engineering problems. *Computer Methods in Applied Mechanics and Engineering*, 193:4417–4438, 2004.

[54] L. Beuth, Z. Więckowski, and P.A. Vermeer. Solution of quasi-static large-strain problems by the material point method. *International Journal for Numerical and Analytical Methods in Geomechanics*, 35:1451–1465, 2011.

[55] Issam Jassim, Dieter Stolle, and Pieter Vermeer. Two-phase dynamic analysis by material point method. *International Journal for Numerical and Analytical Methods in Geomechanics*, 37:2502–2522, 2012.

[56] Ilaria Iaconeta, Antonia Larese, Riccardo Rossi, and Zhiming Guo. Comparison of a material point method and a galerkin meshfree method for the simulation of cohesive-frictional materials. *Materials*, 10:1150, 2017.

[57] S.J. Cummins and J.U. Brackbill. An implicit particle-in-cell method for granular materials. *Journal of Computational Physics*, 180:506–548, 2002.

[58] James Guilkey and Jeffrey Weiss. Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method. *International Journal for Numerical Methods in Engineering*, 57:1323–1338, 2003.

[59] A. Sadeghirad, R. Brannon, and J. Guilkey. Second-order convected particle domain interpolation (cpdi2) with enrichment for weak discontinuities at material interfaces. *International Journal for Numerical Methods in Engineering*, 95:928–952, 2013.

[60] P.C. Wallstedt and J.E. Guilkey. An evaluation of explicit time integration schemes for use with the generalized interpolation material point method. *Journal of Computational Physics*, 227:9628–9642, 2008.

[61] L. Wang, W.M. Coombs, C.E. Augarde, M. Cortis, T.J. Charlton, M.J. Brown, J. Knappet, A. Brennan, C. Davidson, D. Richards, and A. Blake. On the use of domain-based material point methods for problems involving large distortion. *Computer Methods in Applied Mechanics and Engineer-*

ing, 355:1003–1025, 2019.

[62] Michael Andrew Homel, Rebecca Brannon, and James Guilkey. Controlling the onset of numerical fracture in parallelized implementations of the material point method (mpm) with convective particle domain interpolation (cpdi) domain scaling. *International Journal For Numerical Methods in Engineering*, 107:31–48, 2016.

[63] M. Steffen, P.C. Wallstedt, J.E. Guilkey, R.M. Kirby, and M. Berzins. Examination and analysis of implementation choices within the material point method (mpm). *Computer Modeling in Engineering Sciences*, 31:107–127, 2008.

[64] Michael Steffen, Robert M. Kirby, and Martin Berzins. Analysis and reduction of quadrature errors in the material point method (mpm). *International Journal for Numerical Methods in Engineering*, 76:922–948, 2008.

[65] Duan Z. Zhang, Xia Ma, and Paul T. Giguere. Material point method enhanced by modified gradient of shape function. *Journal of Computational Physics*, 230:6379–6398, 2011.

[66] P. Mackenzie-Helnwein, P. Arduino, W. Shin, J.A. Moore, and G.R. Miller. Modeling strategies for multiphase drag interactions using the material point method. *International Journal for Numerical Methods in Engineering*, 86:295–322, 2010.

[67] S.G. Bardenhagen, J. Guilkey, K.M. Roessig, and J.U. Brackbill. An improved contact algorithm for the material point method and application to stress propagation in granular material. *Computer Modeling in Engineering Sciences*, 2:509–522, 2001.

[68] S.G. Bardenhagen. Energy conservation error in the material point method for solid mechanics. *Journal of Computational Physics*, 180:383–403, 2002.

[69] Keita Nakamura, Satoshi Matsumura, and Takaaki Mizutani. Taylor particle-in-cell transfer and kernel correction for material point method. *Computer Methods in Applied Mechanics and Engineering*, 403, 2023.

[70] Fan Zhang, Xiong Zhang, Kam Yim Sze, Yong Liang, and Yan Liu. Improved incompressible material point method based on particle density

correction. *International Journal of Computational Methods*, 15, 2018.

[71] Ming-Jian Li, Yanpng Lian, and Xiong Zhang. An immersed finite element material point (ifemp) method for free surface fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 393, 2022.

[72] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics*, 32, 2013.

[73] S. Leroch, S.J. Eder, G. Ganzenmüller, L.J.S. Murillo, and M. Rodríguez Ripoll. Development and validation of a meshless 3d material point method for simulating the micro-milling process. *Development and validation of a meshless 3D material point method for simulating the micro-milling process*, 262:449–458, 2018.

[74] S. Ma, X. Zhang, and X. Qiu. Comparison study of mpm and sph in modeling hypervelocity impact problems. *International Journal of Impact Engineering*, 36:272–282, 2009.

[75] Zheng Sun, Haiqiao Li, Yong Gan, and Hantao Liu. Material point method and smoothed particle hydrodynamics simulations of fluid flow problems: A comparative study. *Progress in Computational Fluid Dynamics An International Journal*, 18:1, 2018.

[76] S. Ma, X. Zhang, Y. Lian, and X. Zhou. Simulation of high explosive explosion using adaptive material point method. *Computer Modeling in Engineering Sciences*, 39:101, 2009.

[77] M. Urrecha Espluga. *Analysis of Meshfree Methods for Lagrangian Fluid-Structure Interaction*. PhD thesis, Universidad Politécnica de Madrid, Madrid, Spain, 2014.

[78] X. Zhang, K.Y. Sze, and S. Ma. An explicit material point finite element method for hyper-velocity impact. *International Journal For Numerical Methods in Engineering*, 66:689–706, 2006.

[79] Samuel J. Raymond, Bruce Jones, and John R. Williams. A strategy to couple the material point method (mpm) and smoothed particle hydrodynamics (sph) computational techniques. *Computational Particle Mechanics*, 5:49–58, 2018.

[80] J.E. Guilkey, T.B. Harman, and B. Banerjee. Application of material point methods for cutting process simulations. *Computers Structures*, 85:660–674, 2007.

[81] Lin Sun, Sanjay R. Mathur, and Jayathi Y. Murthy. An unstructured finite-volume method for incompressible flows with complex immersed boundaries. *Numerical Heat Transfer, Part B: Fundamentals*, 58:217–241, 2010.

[82] X.X. Cui, X. Zhang, X. Zhou, Y. Liu, and F. Zhang. A coupled finite difference material point method and its application in explosion simulation. *Computer Modeling in Engineering Sciences*, 98:565–599, 2014.

[83] Y.P. Lian, X. Zhang, and Y. Liu. Coupling of finite element method with material point method by local multi-mesh contact method. *Computer Methods in Applied Mechanics and Engineering*, 200:3482–3494, 2011.

[84] Z.P. Chen, X.M. Qiu, X. Zhang, and Y.P. Lian. Improved coupling of finite element method with material point method based on a particle-to-surface contact algorithm. *Computer Methods in Applied Mechanics and Engineering*, 293:1–19, 2015.

[85] Anvar Gilmanov and Sumanta Acharya. A hybrid immersed boundary and material point method for simulating 3d fluid–structure interaction problems. *International Journal for Numerical Methods in Fluids*, 56:2151–2177, 2008.

[86] Yuxin Wang, H.G. Beom, Ming Sun, and Song Lin. Numerical simulation of explosive welding using the material point method. *International Journal of Impact Engineering*, 38:51–60, 2011.

[87] Ravindra Ambati, Xiaofei Pan, Huang Yuan, and Xiong Zhang. Application of material point methods for cutting process simulations. *Computational Materials Science*, 57:102–110, 2012.

[88] Wenqing Hu and Zhen Chen. Model-based simulation of the synergistic effects of blast and fragmentation on a concrete wall using the mpm. *International Journal of Impact Engineering*, 32:2066–2096, 2006.

[89] S. Andersen and L. Andersen. Modelling of landslides with the material-point method. *Computational Geosciences*, 14:137–147, 2010.

[90] Marcelo A. Llano-Serna, Márcio M. Farias, and Dorival M. Pedroso. An

assessment of the material point method for modelling large scale run-out processes in landslides. *Landslides*, 13:1057–1066, 2016.

[91] K. Soga, E. Alonso, A. Yerro, K. Kumar, and S. Bandara. Trends in large-deformation analysis of landslide mass movements with particular emphasis on the material point method. *Géotechnique*, 66:248–273, 2016.

[92] Alba Yerro, Kenichi Soga, and Jonathan Bray. Runout evaluation of oso landslide with the material point method. *Canadian Geotechnical Journal*, 56:1304–1317, 2019.

[93] C. J. Coetzee, P. A. Vermeer, and A. H. Basson. The modelling of anchors using the material point method. *International Journal for Numerical and Analytical Methods in Geomechanics*, 29:879–895, 2005.

[94] C.J. Coetzee, A.H. Basson, and P.A. Vermeer. Discrete and continuum modelling of excavator bucket filling. *Journal of Terramechanics*, 44:177–186, 2007.

[95] N.T.V. Phuong, A.F. van Tol, A.S.K. Elkadi, and A. Rohe. Numerical investigation of pile installation effects in sand using material point method. *Computers and Geotechnics*, 73:58–71, 2016.

[96] Vahid Galavi, Lars Beuth, Bruno Zuada Coelho, Faraz S. Tehrani, Paul Hölscher, and Frits Van Tol. Numerical simulation of pile installation in saturated sand using material point method. *Procedia Engineering*, 175:72–79, 2017.

[97] Elliot James Fern, Alexander Rohe, Kenichi Soga, and Eduardo Alonso. *The Material Point Method for Geotechnical Engineering: A Practical Guide*. CRC Press, 2019.

[98] J.A. Nairn. Material point method calculations with explicit cracks. *Computer Modeling in Engineering Sciences*, 4:649–663, 2003.

[99] Irina Ionescu, James E. Guilkey, Martin Berzins, Robert M. Kirby, and Jeffrey A. Weiss. Simulation of soft tissue failure using the material point method. *Journal of Biomechanical Engineering*, 128:917–924, 2006.

[100] F. Zabala and E.E. Alonso. Progressive failure of aznalco´llar dam using the material point method. *Géotechnique*, 61:795–808, 2015.

[101] Michael A. Homel and Eric B. Herbold. Field-gradient partitioning for

fracture and frictional contact in the material point method. *International Journal for Numerical Methods in Engineering*, 109:1013–1044, 2017.

[102] S. Bardenhagen, A. Brydon, and J. Guilkey. Insight into the physics of foam densification via numerical simulation. *Journal of the Mechanics and Physics of Solids*, 53:597–617, 2005.

[103] J. Nairn. Numerical simulations of transverse compression and densification in wood. *Wood and Fiber Science*, 38:576–591, 2006.

[104] J.H. Lee and D. Huang. Material point method modeling of porous semi-brittle materials. *IOP Conference Series: Materials Science and Engineering*, 10, 2010.

[105] J. E. Guilkey, J. B. Hoying, and J. A. Weiss. Computational modeling of multicellular constructs with the material point method. *Journal of Biomechanics*, 39:2074–2086, 2006.

[106] Patrick G. Hu, Liping Xue, Shaolin Mao, Ramji Kamakoti, Hongwu Zhao, Nagendra Dittakavi, Zhen Wang, Qingding Li, Kan Ni, and Marty Brenner. Material point method applied to fluid-structure interaction (fsi)/aeroelasticity problems. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010.

[107] L.T. Tran, J. Kim, and M. Berzins. Solving time-dependent pdes using the material point method, a case study from gas dynamics. *International Journal for Numerical Methods in Fluids*, 62:709–732, 2010.

[108] Wen-Chia Yang, Pedro Arduino, Gregory R. Miller, and Peter Mackenzie-Helnwein. Smoothing algorithm for stabilization of the material point method for fluid–solid interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 342:177–199, 2018.

[109] Duan Z. Zhang, Qisu Zou, W. Brian VanderHeyden, and Xia Ma. Material point method applied to multiphase flows. *Journal of Computational Physics*, 227:3159–3173, 2008.

[110] Shyamini Kularathna, Weijian Liang, Tianchi Zhao, Bodhinanda Chandra, Jidong Zhao, and Kenichi Soga. A semi-implicit material point method based on fractional-step method for saturated soil. *International Journal for Numerical and Analytical Methods in Geomechanics*, 45:1405–1436, 2021.

[111] Yidong Zhao and Jinhyun Choo. Stabilized material point methods for coupled large deformation and fluid flow in porous materials. *Computer Methods in Applied Mechanics and Engineering*, 362, 2020.

[112] Xiangcou Zheng, Federico Pisanò, Philip J. Vardon, and Michael A. Hicks. Fully implicit, stabilised, three-field material point method for dynamic coupled problems. *Engineering with Computers*, 38:5583–5602, 2022.

[113] C.M. Mast, P. Mackenzie-Helnwein, P. Arduino, G.R. Miller, and W. Shin. Mitigating kinematic locking in the material point method. *Journal of Computational Physics*, 231:5351–5373, 2012.

[114] G. Oger, S. Marrone, D. Le Touzé, and M. de Leffe. Sph accuracy improvement through the combination of a quasi-lagrangian shifting transport velocity and consistent ale formalisms. *Journal of Computational Physics*, 313:76–98, 2016.

[115] Daniel Shigueo Morikawa, Kumpei Tsuji, and Mitsuteru Asai. Corrected ale-isph with novel neumann boundary condition and density-based particle shifting technique. *Journal of Computational Physics: X*, 17, 2023.

[116] Aaron S. Baumgarten and Ken Kamrin. Analysis and mitigation of spatial integration errors for the material point method. *International Journal for Numerical Methods in Engineering*, 124:2449–2497, 2023.

[117] Alexandre Joel Chorin. Numerical solution of the navier-stokes equations. *Mathematics of Computation*, 22:745–762, 1968.

[118] Pooyan Dadvand, Riccardo Rossi, and Eugenio Oñate. An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Archives of Computational Methods in Engineering*, 17:253–297, 2010.

[119] P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela, E. Juanpere, S.R. Idelsohn, and E. Oñate. Migration of a generic multi-physics framework to hpc environments. *Computers Fluids*, 80:301–309, 2013.

[120] Patrick Knupp and Kambiz Salari. *Verification of Computer Codes in Computational Science and Engineering*. Chapman and Hall/CRC, 2002.